



Creating A Single Global Electronic Market

# ***ebXML Business Process Specification Schema Version 1.05(Candidate for Version 2.0)***

**Business Process Project Team**  
**15 July 2002**

## **1 Status of this Document**

This Technical Specification document is a draft. This material fulfills requirements of the ebXML Requirements document. The formatting for this document is based on the Internet Society's Standard RFC format.

***This version:***

***Latest version:***

## **2 ebXML BP/CoreComponents metamodel participants**

### **Project Team Lead**

Brian Hayes (Project Lead), Collaborative Domain

### **Editors**

Pallavi Malu, Intel Corporation: (Project Editor, Issues, XML Schema)

Jean-Jacques Dubray, Eigner: (Editor)

Antoine Lonjon, MEGA International("Transaction Result Computation Diagram")

Ed Buchinski, Treasury Board Secretariat of Canada: (Security)

Arvola Chan, Tibco: (OASIS ebXML Collaboration Protocol Profile and Agreement)

Himagiri (Hima) Mukkamala, Sybase: (OASIS ebXML Messaging, Signal structures)

David Smiley, Mercator: (OASIS ebXML Collaboration Protocol Profile and Agreement)

### **Participants**

Randy Clark, Baker Hughes

Jim Clark, Microsoft

James Bryce Clark, McLure-Moynihan

Karsten Riemer, Sun Microsystems

Larrisa Leybovich, Larrisa Leybovich Consulting

Bob Haugen, Logistical Software

Anders W. Tell, Toolsmiths

Suresh Damodaran, Sterling Commerce

John Yunker, Collaborative Domain

## **3 ebXML Participants**

The authors wish to recognize the following for their significant participation in developing the Business Process Specification Schema, Version 1.01.

Paul Levine, Telcordia

Jim Clark, E2Open - previously Edifecs: (Transaction Semantics)

Cory Casanave, Data Access Technologies: (UML model)

Kurt Kanaskie, Lucent Technologies: (DTD and Examples)

Betty Harvey, Electronic Commerce Connection: (DTD documentation)

Jamie Clark, McLure-Moynihan, Inc.: (Legal aspects)

Neal Smith, Chevron: (Issues Lists, and W3C schema)

John Yunker, Edifecs: (Signal structures)

Karsten Riemer, Sun Microsystems: (Overall Document)

Antoine Lonjon, Mega

J.J. Dubray, Excelon

Bob Haugen, Logistical Software

Bill McCarthy, Michigan State University

Brian Hayes, CommerceOne

Nita Sharma, Netfish

David Welsh, Nordstrom

70	Christopher Ferris, Sun Microsystems
71	Antonio Carrasco, Data Access Technologies
72	

72	<b>4</b>	<b>Table of Contents</b>	
73	1	<i>Status of this Document</i>	<i>i</i>
74	2	<i>ebXML BP/CoreComponents metamodel participants</i>	<i>ii</i>
75	3	<i>ebXML Participants</i>	<i>ii</i>
76	4	<i>Table of Contents</i>	<i>iv</i>
77	5	<i>Introduction</i>	<i>vi</i>
78	5.1	Summary of Contents of Document	vii
79	5.2	Audience	vii
80	5.3	Related Documents	vii
81	5.4	Prerequisites	viii
82	6	<i>Design Objectives</i>	<i>1</i>
83	6.1	Goals/Objectives/Requirements/Problem Description	1
84	6.2	Caveats and Assumptions	1
85	6.2.1	Relationship between <i>ebXML Business Process Specification Schema</i> and UMM	2
86	7	<i>System Overview</i>	<i>4</i>
87	7.1	Key Concepts of the ebXML Business Process Specification Schema	9
88	7.2	How to use the ebXML Business Process Specification Schema	13
89	7.3	How ebXML Business Process Specification Schema is used with other ebXML	
90		specifications	13
91	7.4	How to design collaborations and transactions, re-using at design time	15
92	7.4.1	Packages and Includes	15
93	7.4.2	Substitution Sets	16
94	7.4.3	Specify a Business Transaction and its Business Document Flow	16
95	7.4.4	Specify a Binary Collaboration	25
96	7.4.5	Specify a MultiParty Collaboration	28
97	7.4.6	Specify a Choreography	30
98	7.4.7	The whole model	34
99	7.5	Core Business Transaction Semantics	36
100	7.5.1	Interaction Predictability	36
101	7.5.2	Creating legally binding contracts	39
102	7.5.3	Non-Repudiation	40
103	7.5.4	Authorization security	41
104	7.5.5	Document security	41
105	7.5.6	Reliability	42
106	7.5.7	Parameters required for CPP/CPA	42

107	<b>7.6</b>	<b>Run time Business Transaction semantics</b>	<b>42</b>
108	7.6.1	Timeouts	43
109	7.6.2	Protocol Exceptions	45
110	7.6.3	Computation of the status of a Business Transaction Activity	47
111	<b>7.7</b>	<b>Runtime Collaboration Semantics</b>	<b>49</b>
112	<b>7.8</b>	<b>Where the ebXML Business Process Specification Schema May Be Implemented</b>	<b>49</b>
113			
114	<b>7.9</b>	<b>Guidelines for Business Service Interface Interoperability</b>	<b>50</b>
115	<b>7.10</b>	<b>Collaboration and transaction well-formedness rules</b>	<b>50</b>
116	<b>8</b>	<b><i>ebXML Business Process Specification Schema</i></b>	<b>53</b>
117	<b>8.1</b>	<b>Documentation for the Schema</b>	<b>53</b>
118	8.1.1	Element Attachment	54
119	8.1.2	Element: AttributeSubstitution	55
120	8.1.3	Element: BinaryCollaboration	56
121	8.1.4	Element: BusinessDocument	57
122	8.1.5	Element: BusinessPartnerRole	58
123	8.1.6	Element: BusinessTransaction	59
124	8.1.7	Element: BusinessTransactionActivity	59
125	8.1.8	Element: CollaborationActivity	61
126	8.1.9	Element: ConditionExpression	62
127	8.1.10	Element: Documentation	62
128	8.1.11	Element: DocumentEnvelope	63
129	8.1.12	Element: DocumentSubstitution	64
130	8.1.13	Element: Failure	64
131	8.1.14	Element: Fork	65
132	8.1.15	Element: Include	66
133	8.1.16	Element: Role	67
134	8.1.17	Element: Join	67
135	8.1.18	Element: MultiPartyCollaboration	68
136	8.1.19	Element: Package	69
137	8.1.20	Element: Performs	70
138	8.1.21	Element: ProcessSpecification	71
139	8.1.22	Element: RequestingBusinessActivity	72
140	8.1.23	Element: RespondingBusinessActivity	73
141	8.1.24	Element: Start	75
142	8.1.25	Element: SubstitutionSet	75
143	8.1.26	Element: Success	76
144	8.1.27	Element: Transition	77
145	8.1.28	complexType BusinessAction	78
146	8.1.29	complexType BusinessActivity	79
147	8.1.30	complexType RoleType	80
148	8.1.31	attributeGroup documentSecurity	80

149	8.1.32	attributeGroup name	81
150	<b>8.2</b>	<b>XML to UML cross-reference</b>	<b>81</b>
151	<b>8.3</b>	<b>Scoped Name Reference</b>	<b>83</b>
152	<b>8.4</b>	<b>Sample XML document against above Schema</b>	<b>84</b>
153	<b>9</b>	<b><i>Business signal structures</i></b>	<b>84</b>
154	9.1.1	Signal Schema	84
155	9.1.2	ReceiptAcknowledgment Signal Schema	85
156	9.1.3	AcceptanceAcknowledgement Signal Schema	86
157	9.1.4	Exception Signal Schema	86
158	<b>10</b>	<b><i>EDI support</i></b>	<b>87</b>
159	<b>11</b>	<b><i>Production Rules</i></b>	<b>87</b>
160	<b><i>Appendix A: Sample XML Business Process Specification</i></b>		<b>89</b>
161	<b><i>Appendix B: Sample XML Signal</i></b>		<b>92</b>
162	<b>12</b>	<b><i>References</i></b>	<b>93</b>
163	<b>13</b>	<b><i>Disclaimer</i></b>	<b>93</b>
164	<b>14</b>	<b><i>Contact Information</i></b>	<b>94</b>

## 5 Introduction

### EXECUTIVE SUMMARY

---

The ebXML Business Process Specification Schema provides a standard framework by which business systems may be configured to support execution of business collaborations consisting of business transactions. It is based upon prior UN/CEFACT work, specifically the metamodel behind the UN/CEFACT Modeling Methodology (UMM) defined in the N090R10 specification.

The Specification Schema supports the specification of Business Transactions and the choreography of Business Transactions into Business Collaborations. Each Business Transaction can be implemented using one of many available standard patterns. These patterns determine the actual exchange of Business Documents and business signals between the partners to achieve the required electronic commerce transaction.

The current version of the specification schema addresses collaborations between two parties (Binary Collaborations) as well as collaboration involving more than two business partners (Multiparty Collaborations) as a synthesis of binary collaborations.

It is anticipated that a subsequent version will address additional features such as the semantics of economic exchanges and contracts, more complex multi-party choreography, and context based content.

## 5.1 Summary of Contents of Document

This document describes the ebXML Specification Schema.

This document describes the Specification Schema in its UML form and provides the corresponding XML Schema which every Specification Schema instance must conform to.

The document first introduces general concepts and semantics, then applies these semantics in a detail discussion of each part of the model. The document then specifies all elements in the UML form, and then in XML form.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [Bra97].

## 5.2 Audience

The primary audience is business process analysts. We define a business process analyst as someone who interviews business people and as a result documents business processes in unambiguous syntax.

An additional audience is designers of business process definition tools who need to specify the conversion of user input in the tool into the XML representation of the Specification Schema.

The audience is not business application developers.

## 5.3 Related Documents

As mentioned above, other documents provide detailed definitions of some of the components of the ebXML Business Process Specification Schema and of their inter-relationship. They include ebXML Specifications on the following topics:

- ebXML Technical Architecture Specification, version 1.04
- ebXML Core Components Dictionary, version 1.04
- ebXML Naming Convention for Core Components, version 1.04
- ebXML Collaboration-Protocol Profile and Agreement Specification V2.0
- ebXML Business Process and Business Information Analysis Overview, version 1.0
- ebXML Business Process Analysis Worksheets & Guidelines, version 1.0
- ebXML E-Commerce Patterns, version 1.0
- ebXML Catalog of Common Business Processes, version 1.0
- ebXML Message Service Specification V2.0

- 220 • UN/CEFACT Modeling Methodology (UMM) as defined in the N090R10  
221 specification

## 222 **5.4 Prerequisites**

223 It is assumed that the audience will be familiar with or have knowledge of the following  
224 technologies and techniques:

- 225 • Business process modeling techniques and principles
- 226 • The UML syntax and semantics
- 227 • The Extensible Markup Language (XML)



## 6 Design Objectives

### 6.1 Goals/Objectives/Requirements/Problem Description

ebXML Business Process Specification Schema models describe interoperable business processes that allow business partners to collaborate. These models must be executed by software components that collaborate on behalf of the business partners.

The goal of the ebXML Business Process Specification Schema is to provide the bridge between e-business process modeling and specification of e-business software components.

The ebXML Business Process Specification Schema provides for the nominal set of specification elements necessary to specify a collaboration between business partners, and to provide configuration parameters for the partners' runtime systems in order to execute that collaboration between a set of e-business software components.

A specification created against the ebXML Business Process Specification Schema is referred to as an ebXML Specification.

The *ebXML Specification Schema* is available as an XML Schema (<http://www.w3.org/2001/XMLSchema>). This document also provides a UML description of the schema.

The UML version of the *ebXML Business Process Specification Schema* is merely a UML Class Diagram. It is not intended for the direct creation of ebXML Business Process Specifications. Rather, it is a self-contained statement of all the specification elements and relationships required to be able to create an ebXML compliant Business Process Specification. Any methodologies and/or metamodels used for the creation of ebXML compliant Business Process Specifications must at minimum support these elements and relationships.

The XML Schema provides the specification for XML based instances of ebXML Specifications.

The UML and XML based representations of the *ebXML Business Process Specification Schema* are unambiguously mapped to each other.

### 6.2 Caveats and Assumptions

This specification is designed to specify the run time aspects of a business collaboration.

It is not intended to incorporate a methodology, and does not directly prescribe the use of a methodology. However, if a methodology is to be used, it is recommended that it be UN/CEFACT Modeling Methodology (UMM).

The *ebXML Business Process Specification Schema* does not by itself define Business Documents Structures. It is intended to work in conjunction with already existing Business Document definitions, and/or the document metamodel defined by the ebXML Core Components specifications.

## 6.2.1 Relationship between *ebXML Business Process Specification Schema* and UMM

The UN/CEFACT Modeling Methodology (UMM) is a methodology for business process and information modeling.

This section describes the relationship between UMM and the *ebXML Business Process Specification Schema*.

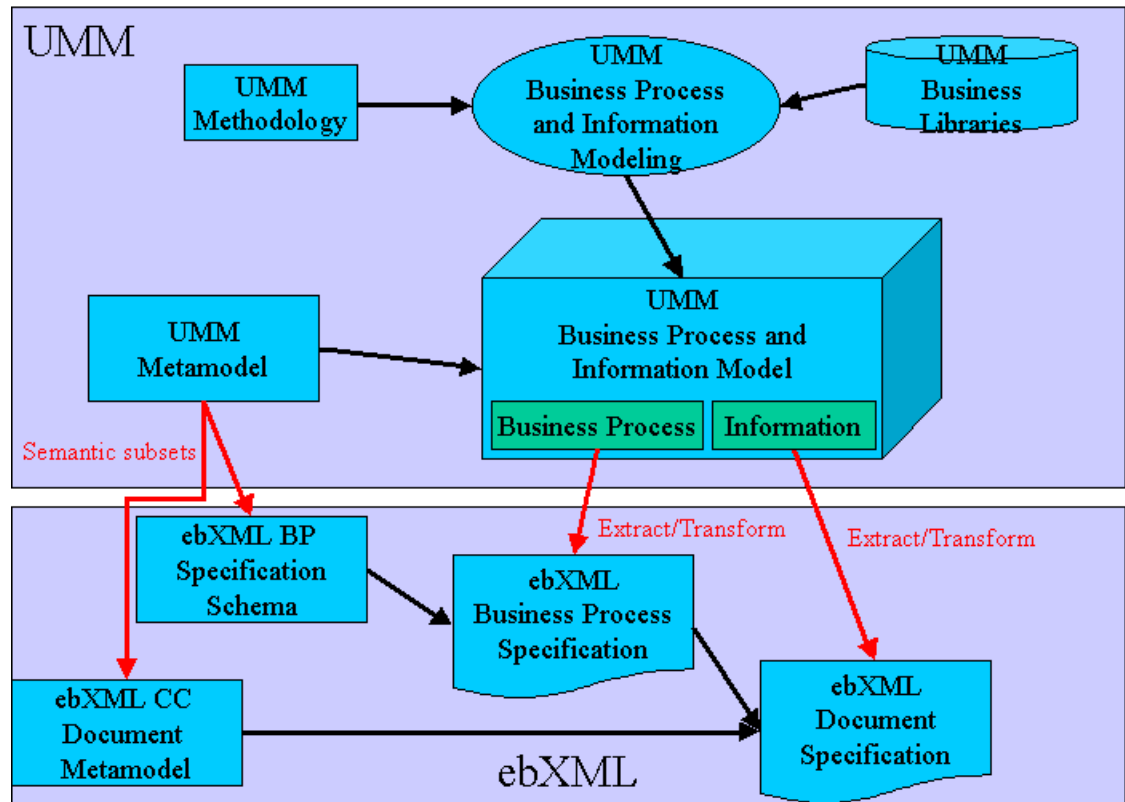
The UMM Meta Model is a description of business semantics that allows Trading Partners to capture the details for a specific business scenario (a Business Process) using a consistent modeling methodology. A Business Process describes in detail how Trading Partners take on shared roles, relationships and responsibilities to facilitate interaction with other Trading Partners. The interaction between roles takes place as a choreographed set of Business Transactions. Each Business Transaction is expressed as an exchange of electronic Business Documents. The sequence of the exchange is determined by the Business Process, and by messaging and security considerations. Business Documents are composed from re-useable Business Information Objects. At a lower level, Business Processes can be composed of re-useable Common Business Processes, and Business Information Objects can be composed of re-useable Core Components. Common Business Processes and Business Information Objects reside in a UMM Business Library.

The UMM Meta Model supports a set of Business Process viewpoints that provide a set of semantics (vocabulary) for each viewpoint and forms the basis for specification of the semantics and artifacts that are required to facilitate business process and information integration and interoperability. Using the UMM methodology and the UMM metamodel, the user may thus create a complete Business Process and Information Model. This model contains more information than what is required for configuring ebXML compliant software. Also the model is syntax independent and not directly interpretable by ebXML compliant software.

The *ebXML Business Process Specification Schema* provides an additional view of the UMM metamodel. This subset is provided to support the direct specification of the nominal set of elements necessary to configure a runtime system in order to execute a set of ebXML business transactions. By drawing out modeling elements from several of the other views, the *ebXML Business Process Specification Schema* forms a semantic subset of the UMM Meta Model. Using the *ebXML Business Process Specification Schema* the user may thus create a Business Process Specification that contains only the information required to configure ebXML compliant software.

It is expected that ebXML compliant software will be configured with XML instances conforming to the *ebXML Business Process Specification Schema*.

The relationship between the UMM Meta Model and the *ebXML Business Process Specification Schema* is shown in Figure 1.

Figure 1. UMM Metamodel and *ebXML Business Process Specification Schema*

Using the UMM methodology, and drawing on content from the UMM Business Library a user may create complete Business Process and Information Model conforming to the UMM metamodel.

Since the *ebXML Business Process Specification Schema* is a semantic subset of the UMM metamodel, the user may then in an automated fashion extract from the Business Process and Information Model the required set of elements and relationships, and transform them into an *ebXML Business Process Specification* conforming to the *ebXML Business Process Specification Schema*.

Likewise, since the *ebXML CC document metamodel* is aligned with the UMM Metamodel, the user may then in an automated fashion extract from the Business Process and Information Model the required set of elements and relationships, and transform them into an *ebXML document model* conforming to *ebXML Core Component specifications*.

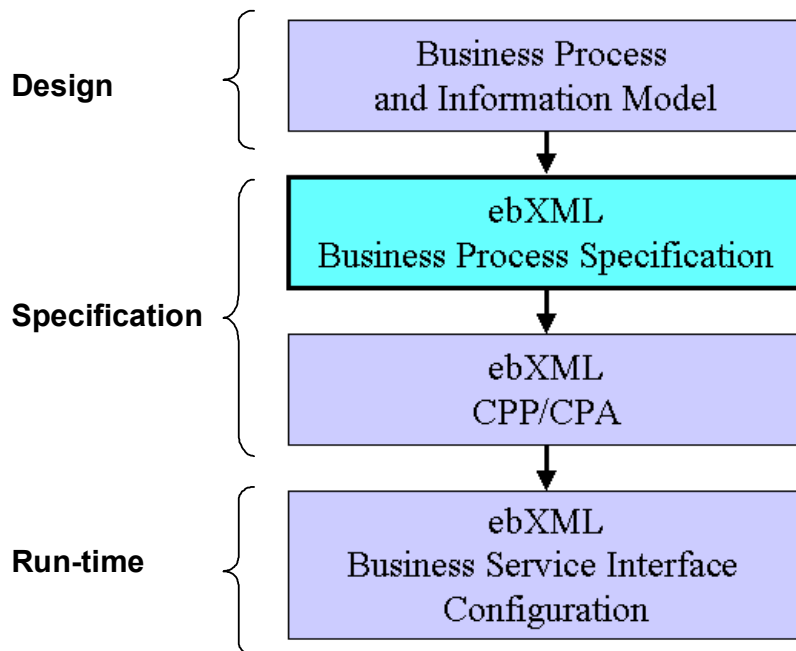
The UMM methodology is not part of the formal set of *ebXML specifications*.

Likewise, the UMM metamodel in its entirety is not part of the formal set of *ebXML specifications*. Only the semantic subset represented by the *ebXML Business Process Specification Schema* and CC are parts of the formal set of *ebXML specifications*.

The remainder of this document focuses on the *ebXML Business Process Specification Schema* and Business Process Specifications created against it. It is understood that proper Business Process and Information Modeling may have taken place prior to beginning the activity of creating a Business Process Specification.

## 7 System Overview

The *ebXML Business Process Specification Schema* provides a standard framework for business process specification. As such, it works with the ebXML Collaboration Protocol Profile (CPP) and Collaboration Protocol Agreement (CPA) specifications to bridge the gap between Business Process Modeling and the configuration of ebXML compliant e-commerce software, e.g. an ebXML Business Service Interface, as depicted in Figure 2.



**Figure 2: Business Process Specification and Business Service Interface Configuration**

Using Business Process Modeling, a user may create a complete Business Process and Information Model.

Based on this Business Process and Information Model and using the *ebXML Business Process Specification Schema* the user will then extract and format the nominal set of elements necessary to configure an ebXML runtime system in

352 order to execute a set of ebXML business transactions. The result is an ebXML  
353 *Business Process Specification*.

354 Alternatively the ebXML *Business Process Specification* may be created directly,  
355 without prior explicit business process modeling.

356 An ebXML *Business Process Specification* contains the specification of Business  
357 Transactions and the choreography of Business Transactions into Business  
358 Collaborations.

359 This ebXML *Business Process Specification* is then the input to the formation of  
360 ebXML trading partner Collaboration Protocol Profiles and Collaboration Protocol  
361 Agreements.

362 These ebXML trading partner Collaboration Protocol Profiles and Collaboration  
363 Protocol Agreements in turn serve as configuration files for ebXML Business  
364 Service Interface software. The ebXML Business Interface Software represents  
365 any ebXML compliant component, which is able to be, configured from an ebXML  
366 Specification Schema, a CPP and a CPA.

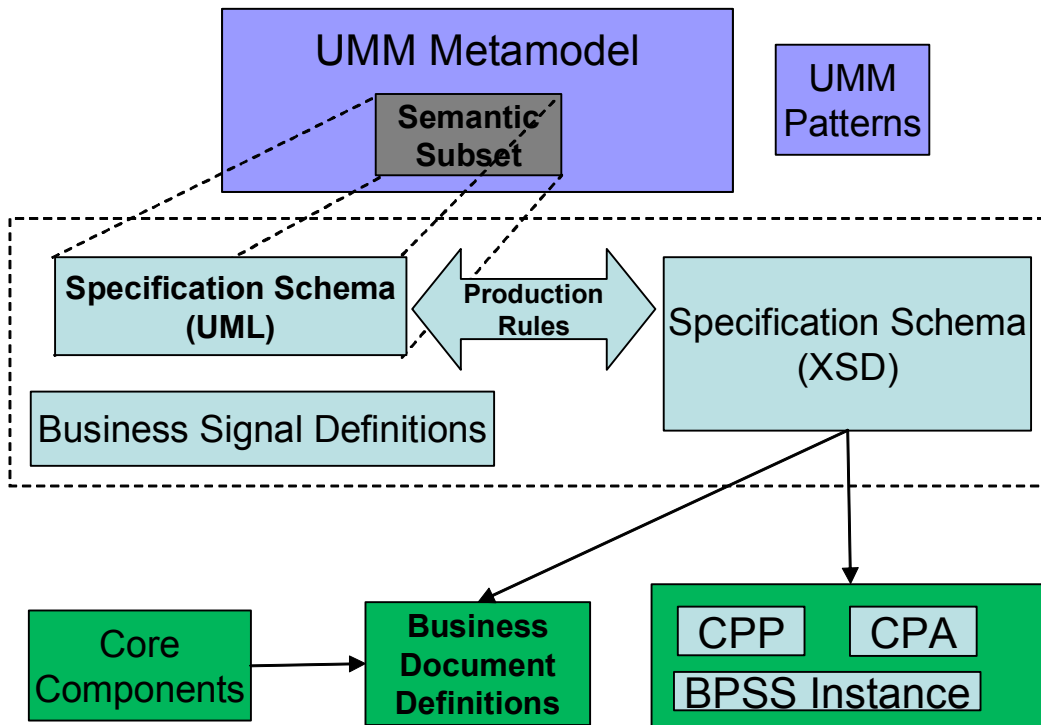
367 The architecture of the ebXML *Business Process Specification Schema* consists  
368 of the following functional components:

- 369 • UML version of the *Business Process Specification Schema*
- 370 • XML version of the *Business Process Specification Schema*
- 371 • Production Rules defining the mapping from the UML version of the  
372 *Business Process Specification Schema* to the XML version
- 373 • Business Signal Definitions

374

375 Together these components allow you to fully specify all the run time aspects of a  
376 business process model.

These components are shown (inside the dotted box) in figure 3 below.



**Figure 3: Relationship of *ebXML Business Process Specification Schema* to UMM, CPP/CPA and Core Components**

The following provides a description of each of the components in the *ebXML Business Process Specification Schema* and their relationship to UMM, and *ebXML CC* and CPP/CPA:

### ***UML Representation of Business Process Specification Schema***

The UML representation of the *ebXML Business Process Specification Schema* is a semantic subset of the metamodel behind UMM as specified in UN/CEFACT TMWG's N090R10

N090R10 is as of this writing not yet approved by UN/CEFACT. It is the intent to keep the *ebXML Business Process Specification Schema* and the UN/CEFACT TMWG's N090 semantically aligned.

The UML version of the *ebXML Business Process Specification Schema* is merely a UML Class Diagram. It is not intended for the direct creation of *ebXML Business Process Specifications*. Rather, it is a self-contained statement of all the specification elements and relationships required to be able to create an *ebXML compliant Business Process Specification*.

## ***XML Schema representation of Business Process Specification Schema***

The corresponding XML Schema representation of the ebXML *Business Process Specification Schema* provides the specification for XML based instances of ebXML Business Process Specifications, and as a target for production rules from other representations. Thus, a user may either create a *Business Process Specification* directly as an XML document, or may chose to use some other means of specification first and then apply production rules to arrive at the XML document version.

Any methodologies and/or metamodels used for the creation of ebXML compliant Business Process Specifications must at minimum support the production of the elements and relationships contained in the XML representation of the ebXML *Business Process Specification Schema*.

This XML Schema definition is isomorphic to the UML representation of the ebXML *Business Process Specification Schema*.

## ***UMM Business Process Interaction Patterns***

Any ebXML Business Service Interface software components should be able to be configured to execute the business processes specified in a *Business Process Specification*. They do so by exchanging ebXML messages and business signals.

Each Business Transaction can be implemented using one of many available standard patterns. These patterns determine the actual exchange of messages and business signals between the partners to achieve the required electronic commerce transaction.

The Business Transaction Interaction Patterns set forth in Chapter 8 of the UMM N090R10 document illustrate recommended permutations of message sequences as determined by the type of business transaction defined and the timing policies specified in the transactions.

While the UMM patterns themselves are not part of the ebXML specifications, all the security and timing parameters required to express the pattern properties are provided as attributes of elements in the ebXML *Business Process Specification Schema*.

## ***Business Signal Definitions***

Business signals are application level documents that 'signal' the current state of the business transaction. These business signals have specific business purpose and are separate from lower protocol and transport signals as specified in the ebXML Message Service Specification.

However, the structures of ebXML business signals are 'universal' and do not vary from transaction to transaction. Thus, they can be defined once and for all as part of the ebXML *Business Process Specification* itself.

439 The Business Process Specification provides both the choreography of business  
440 signals, and the structure definition of the business payload of a business signal.  
441 The ebXML Message Service Specification signal structures provide business  
442 service state alignment infrastructure, including unique message identifiers and  
443 digests used to meet the basic process alignment requirements. The business  
444 signal payload structures provided herein are optional and normative and are  
445 intended to provide business and legal semantics to the business signals.  
446 A Schema is provided for each of the possible business signals.

## 447 **Production Rules**

448 A set of production rules is provided, defining the mapping from the UML version  
449 of the ebXML *Business Process Specification Schema* to the XML version.

450 The primary purpose for these production rules is to govern the one-time  
451 generation of the Schema representation of the ebXML *Business Process*  
452 *Specification Schema* from the UML Class Diagram version of the ebXML  
453 *Business Process Specification Schema*.

454 The Class Diagram representation of *Business Process Specification Schema* is  
455 not intended for the direct creation of ebXML Business Process Specifications.  
456 However, if a *Business Process Specification* were in fact (programmatically)  
457 created as an instance of this class diagram, the production rules would also  
458 apply for its conversion into a Schema conformant XML document.

459 Separately, it is expected that a set of production rules will be constructed for the  
460 production of an XML version of an ebXML *Business Process Specification* from  
461 a set of UML diagrams constructed through the use of UMM.

462 An instance of the UML Class Diagram version of the ebXML *Business Process*  
463 *Specification Schema* will through the application of its production rules produce  
464 an XML Specification Document that is analytically, semantically and functionally  
465 equivalent to one arrived at by modeling the same subset through the use of  
466 UMM and its associated production rules.

## 467 **Relationship to CPP/CPA**

468 A *Business Process Specification* is, along with protocol specifications, the object  
469 of the agreement between two parties. The *Business Process Specification* is  
470 therefore incorporated with or referenced by ebXML trading partner Collaboration  
471 Protocol Profiles (CPP) and Collaboration Protocol Agreements (CPA). Each  
472 CPP declares its support for one or more Roles within the *Business Process*  
473 *Specification*. A Business Process Specification is also a machine interpretable  
474 specification needed for an ebXML Business Service Interface, which will enforce  
475 its definition at run-time. The CPP profiles and CPA agreements contain further  
476 technical parameters resulting in a full specification of the run-time software at  
477 each trading partner.

## 478 **Relationship to CC**



479 The *Business Process Specification Schema* does not by itself support the  
480 definition of Business Documents. Rather, a *Business Process Specification*  
481 merely points to the definition of Business Documents. Such definitions may  
482 either be XML based, or – as attachments – may be any other structure, or  
483 completely unstructured. XML based Business Document Specifications may be  
484 based on the ebXML Core Components specifications.

### 485 ***Relationship to ebXML Message Service Specification***

486 The Business Process Specification Schema will provide choreography of  
487 business messages and signals. The ebXML Message Service Specification  
488 provides the infrastructure for message / signal identification, typing, and  
489 integrity; as well as placing any one message in sequence with respect to other  
490 messages in the choreography.

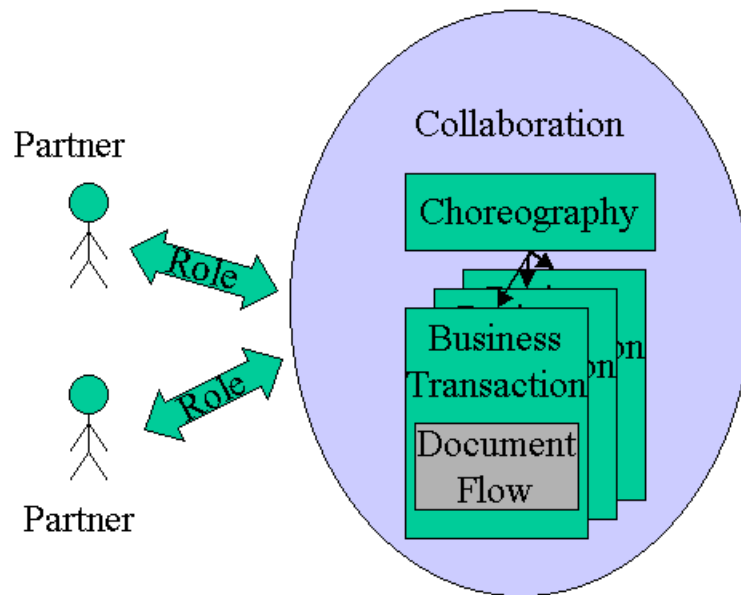
491

## 492 **7.1 Key Concepts of the ebXML Business Process Specification** 493 **Schema**

494  
495 The ebXML *Business Process Specification Schema* provides the semantics,  
496 elements, and properties necessary to define business collaborations.

497 A business collaboration consists of a set of roles collaborating through a set of  
498 choreographed transactions by exchanging business documents.

499 These basic semantics of a business collaboration are shown in Figure 4.



**Figure 4. Basic Semantics of a business collaboration**

Two or more business partners participate in the business collaboration through roles. The roles interact with each other through Business Transactions. The business transactions are sequenced relative to each other in Choreography. Each Business Transaction consists of one or two predefined Business document flows. One or more Business Signals may additionally support a Business Transaction.

The following section describes the concepts of a Business Collaboration, a Business Transaction, a Business document flow, and Choreography

#### 1. Business Collaborations

A business collaboration is a set of Business Transactions between business partners. Each partner plays one or more roles in the collaboration.

The ebXML *Business Process Specification Schema* supports two levels of business collaborations, Binary Collaborations and Multiparty Collaborations.

Binary Collaborations are between two roles only.

Multiparty Collaborations are between more than two roles, but such Multiparty Collaborations are always synthesized from two or more Binary Collaborations. For instance if Roles A, B, and C collaborate and all parties interact with each other, there will be a separate Binary Collaboration between A and B, one between B and C, and one between A and C. The Multiparty Collaboration will be the synthesis of these three Binary Collaborations.

Binary Collaborations are expressed as a set of Business Activities between the two roles. Each Business Activity reflects a state in the collaboration. The Business Activity can be a Business Transaction Activity, i.e. the activity of conducting a single Business Transaction, or a Collaboration Activity, i.e. the activity of conducting another Binary Collaboration. An example of the former is the activity of placing a purchase order. An example of the latter is the activity of negotiating a contract. In either case the activities can be choreographed relative to other activities as per below.

The ability of a Binary Collaboration to have activities that in effect are executing other Binary Collaborations is the key to recursive compositions of Binary Collaboration, and to the re-use of Binary Collaborations. An activity, whether it is a Business Transaction Activity or a Collaboration Activity represents the usage of a definition within a Binary Collaboration Specification. For instance, a Business Transaction is defined once and for all, but could appear several times – as a Business Transaction Activity -, sometimes even with opposite roles, within the same binary collaboration definition.

In essence each Binary Collaboration is a re-useable protocol between two roles.

## 2. Business Transactions

A Business Transaction is the atomic unit of work in a trading arrangement between two business partners. A Business Transaction is conducted between two parties playing opposite roles in the transaction. The roles are always a requesting role and a responding role. They are not specific roles like buyer or seller. These roles will be specified at the Business Transaction Activity level, when the Business Transaction definition is used for a specific purpose.

Like a Binary Collaboration, a Business Transaction is a re-useable protocol between two roles. The way it is re-used is by referencing it from a Binary Collaboration through the use of a Business Transaction Activity as per above. In a Business Transaction Activity the roles of the Binary Collaboration are assigned to the execution of the Business Transaction.

Unlike a Binary Collaboration, however, the Business Transaction is atomic; it cannot be decomposed into lower level Business Transactions.

A Business Transaction is a very specialized and very constrained protocol, in order to achieve very precise and enforceable transaction semantics. These semantics are expected to be enforced by the software

564 managing the transaction, i.e. an ebXML Business Service Interface (BSI)  
565 software component.

566 A Business Transaction will always either succeed or fail both from a  
567 protocol and a business perspective. If it succeeds from both perspective  
568 it may be designated as legally binding between the two partners, or  
569 otherwise govern their collaborative activity. If it fails it is null and void,  
570 and each partner must relinquish any mutual claim established by the  
571 transaction. In addition, if it fails from protocol perspective, each party  
572 must synchronize their state to the state prior the start of the transaction.  
573 In case of a business failure, the state has already been synchronized  
574 and it is the duty of each application to take the proper actions.

### 575 3. Business Document flows

576 A business transaction is realized as Business Document flows between  
577 the requesting and responding roles. There is always a requesting  
578 Business Document, and optionally a responding Business Document,  
579 depending on the desired transaction semantics, e.g. one-way notification  
580 vs. two-way conversation.

581 Actual document definition is achieved using the ebXML core component  
582 specifications, or by some methodology external to ebXML but resulting in  
583 Schema definition (XSD or DTD) that an ebXML *Business Process*  
584 *Specification* can point to.

### 585 4. Choreography

586 The Business Transaction Choreography describes the ordering and  
587 transitions between business transactions or sub collaborations within a  
588 binary collaboration. For example, in a UML tool this could be done using  
589 a UML activity diagram. The choreography is described in the ebXML  
590 *Business Process Specification Schema* using activity diagram concepts  
591 such as start state, completion state, activities, synchronizations,  
592 transitions between activities, and guards on the transitions. This  
593 document does not specify a graphical representation of collaboration.

### 594 5. Patterns

595 The ebXML *Business Process Specification Schema* provides a set of  
596 unambiguous semantics within which to specify transactions and  
597 collaborations. Within these semantics the user community has flexibility  
598 to specify an infinite number of specific transactions and collaborations.  
599 The use of predefined patterns combines this flexibility with a consistency  
600 that facilitates faster design, faster implementation, and enables generic  
601 processing.

602 A set of predefined transaction interaction patterns, defining common  
603 combinations of transaction interaction parameter settings can be found  
604 in UMM.

605 While the UMM transaction interaction patterns themselves are not part of  
606 the ebXML specifications, all the security and timing parameters required

to express the pattern properties are provided as attributes of elements in the *Business Process Specification Schema*.

It is also anticipated that patterns for collaboration choreographies will emerge. An example of such a pattern is in the ebXML E-Commerce Patterns.

Re-use, recursion, and patterns are among the key concepts of the ebXML *Business Process Specification Schema*. The following section will illustrate these key concepts.

## **7.2 How to use the ebXML Business Process Specification Schema**

The ebXML *Business Process Specification Schema* should be used wherever ebXML compliant software is being specified to execute Business Collaborations. The generic term for such software is a Business Service Interface (BSI).

The ebXML *Business Process Specification Schema* is used to specify the business process related configuration parameters for configuring a BSI to execute these collaborations.

This section discusses

- How the ebXML *Business Process Specification Schema* fits in with other ebXML specifications.
- How to use the ebXML *Business Process Specification Schema* at design time, either for specifying brand new collaborations and transactions, or for re-using existing ones.
- How to specify core transaction semantics and parameters needed for a Collaboration-Protocol Profile and Agreement (CPP/CPA).
- Run-time transaction and collaboration semantics that the ebXML *Business Process Specification Schema* specifies and the Business Service Interface (BSI) is expected to manage.

## **7.3 How ebXML Business Process Specification Schema is used with other ebXML specifications**

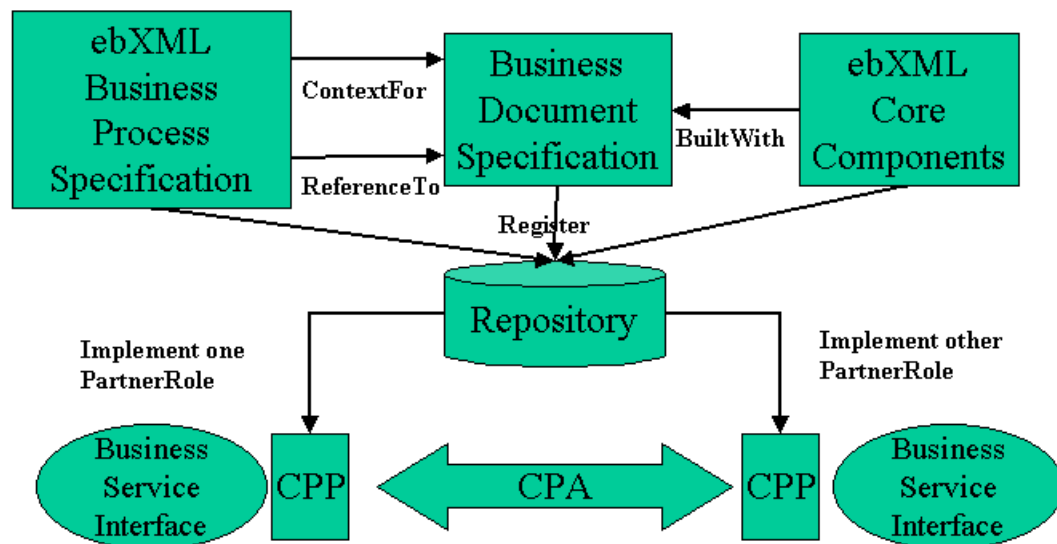
The ebXML *Business Process Specification Schema* provides the semantics, elements, and properties necessary to define Business Collaborations.

A collaboration consists of a set of roles collaborating through a set of choreographed transactions by exchanging Business Documents.

As shown in Figure 5, Business Documents are defined at the intersection between the Business Process Specification and the ebXML Core Component specifications. A Business Process Specification will reference, but not define, a set of required Business Documents. At ebXML Business Documents are either defined by some external document specification, or assembled directly or indirectly from lower level information structures called core components. The

assembly is based on a set of contexts, many of which are provided by the business processes, i.e. collaborations that use the documents in their document flows.

The combination of the business process specification and the document specification become the basis against which partners can make agreements on conducting electronic business with each other.



654

655 **Figure 5: ebXML Business Process Specification Schema and other ebXML Specifications**

656

657 The user will extract and transform the necessary information from an existing  
 658 Business Process and Information Model. Associated production rules could aid  
 659 in creating an XML representation of a *Business Process Specification*.

660 Alternatively a user would use an XML based tool to produce the XML  
 661 representation directly. Production rules could then aid in converting into XML, so  
 662 that it could be loaded into a UML tool, if required.

663 In either case, the XML representation of the *Business Process Specification*  
 664 gets stored in the ebXML repository and registered in the ebXML registry for  
 665 future retrieval. The *Business Process Specification* would be registered using  
 666 classifiers derived during its design.

667 When implementers want to establish trading partner Collaboration Protocol  
668 Profile and Agreement the *Business Process Specification* XML document, or the  
669 relevant parts of it, are simply imbedded in or referenced by the CPP and CPA  
670 XML documents. ebXML CPP and CPA documents can only reference ebXML  
671 *Business Process Specifications* and only XML versions thereof.

672 Guided by the CPP and CPA specifications the resulting XML document then  
673 becomes the configuration file for one or more Business Service Interfaces (BSI),  
674 i.e. the software that will actually manage either partner's participation in the  
675 collaboration.

## 676 **7.4 How to design collaborations and transactions, re-using at** 677 **design time**

678

679 This section describes the ebXML *Business Process Specification Schema*  
680 modeling relationships by building a complete Multiparty Collaboration from the  
681 bottom up, as follows:

- 682 1. Specify a Business Transaction
- 683 2. Specify the Business Document flow for a Business Transaction
- 684 3. Specify a Binary Collaboration re-using the Business Transaction
- 685 4. Specify a Choreography for the Binary Collaboration
- 686 5. Specify a higher level Binary Collaboration re-using the lower level Binary  
687 Collaboration
- 688 6. Specify a Multiparty Collaboration re-using Binary Collaborations

689 Although this section, for purposes of introduction, discusses the specification of  
690 collaboration from the bottom up, the ebXML *Business Process Specification*  
691 *Schema* very much is intended for specifying collaborations from the top down,  
692 re-using existing lower level content as much as possible.

693 The constructs listed above support the specification of fairly complex multi party  
694 collaborations. However, an ebXML compliant Business Process Specification  
695 may be as simple as a single Binary Collaboration referencing a single Business  
696 Transaction. This involves only numbers 1 through 3 above. In other words,  
697 Higher-level Binary Collaborations, Multi-party Collaborations and choreography  
698 expressions are not required for ebXML Business Process Specification  
699 compliance.

700

### 701 **7.4.1 Packages and Includes**

702 All model elements of this specification are defined within the context of a  
703 package. Packages may contain other package, therefore defining a hierarchy of  
704 packages.

705 A package defines the namespace of the elements inside it. You cannot have  
706 two model elements with the same name within the same package. Model  
707 element names can be qualified with the package using the Java notation:

708 Business Transaction.Order Entry.Process Purchase Order

709 Which means that the Process Purchase Order business transaction is defined  
710 within the package Order Entry, which is itself, defined within the Business  
711 Transaction package. Note that there is no ambiguity in using spaces within the  
712 names of packages or model elements.

713 If a model element in package Order Entry needs to name something in a  
714 package called Billing, it must include this package to make its elements visible  
715 to its own model elements. Unlike an import, include requires that all model  
716 elements from the Billing package be fully qualified. So if we want to designate  
717 the Invoice business document within the Order Entry.Process Purchase Order  
718 transaction we need to refer to the Billing.Invoice document, assuming it is  
719 defined in the Business Transaction.Billing package.

## 720 7.4.2 Substitution Sets

721 There is a requirement for Business specifications that are less coupled to  
722 technology and business details, such as specific document formats and  
723 structures and timing parameters. Substitution sets support the capability to take  
724 a generic business process and specialize it for a specific use. For example, an  
725 ordering process may be very generic but a specific use of that process may  
726 require specific document capabilities that go beyond the generic.

727 A substitution set is placed in the more specific process specification and  
728 replaces or makes more explicit document definition references and attribute  
729 values. A Substitution Set is a container for one or more AttributeSubstitution  
730 and/or DocumentSubstitution elements. The entire SubstitutionSet specifies  
731 document or attribute values that should be used in place of some documents  
732 and attribute values in an existing process specification.

733

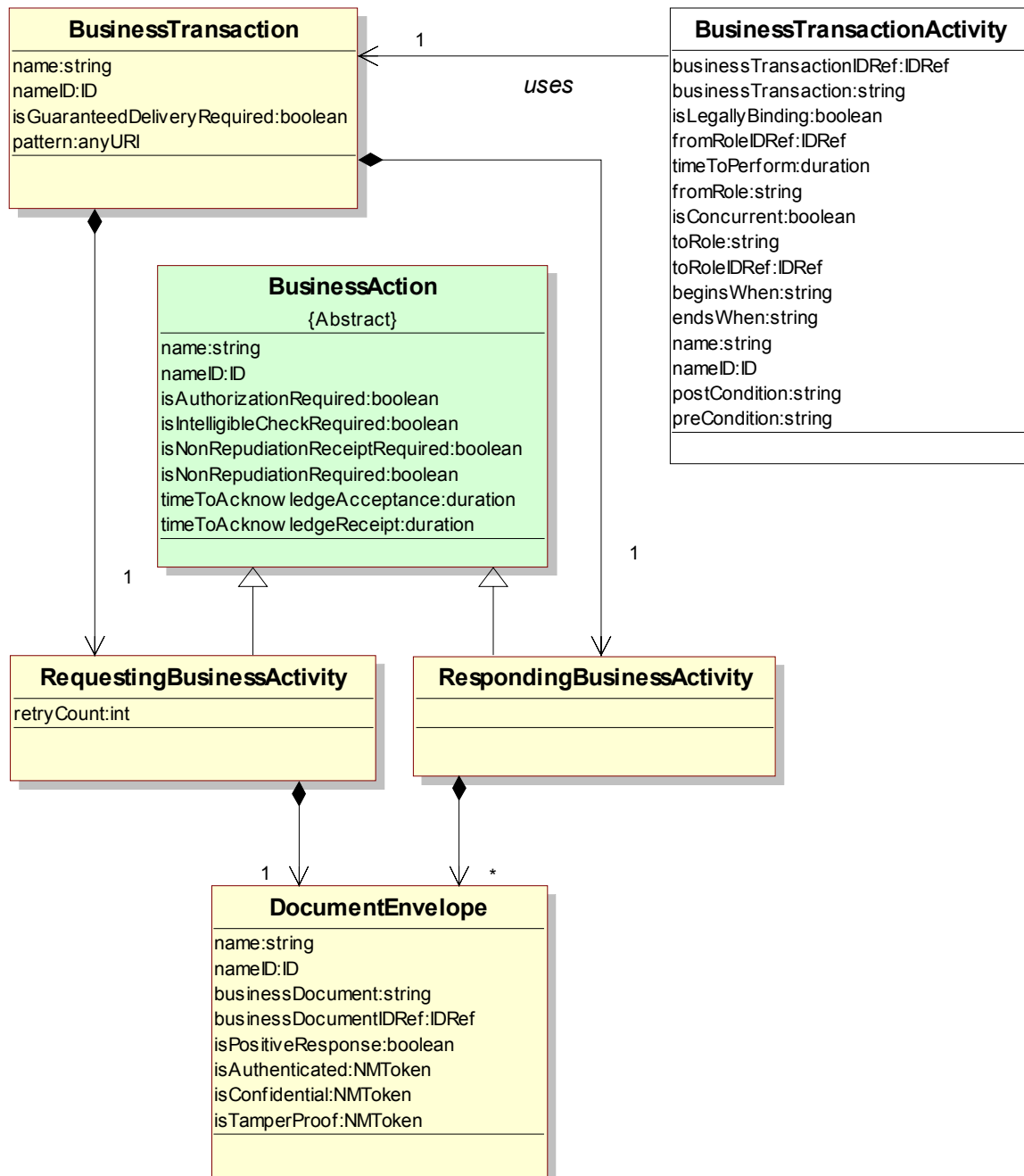
734

## 735 7.4.3 Specify a Business Transaction and its Business Document 736 Flow

737

738 Figure 6 illustrates a business transaction.





741 **Figure 6. UML Diagram of a Business Transaction**

### 742 7.4.3.1 Key Semantics of a Business Transaction

743 A Business Transaction is the atomic unit of work in a trading  
 744 arrangement between two business partners.  
 745  
 746

A business transaction consists of a Requesting Business Activity, a Responding Business Activity, and one or two document flows between them. One or more Business Signals that govern the use and meaning of acknowledgements and related matters in the transaction may additionally support a Business Transaction.

Implicitly there is a requesting role performing the Requesting Business Activity and a responding role performing the Responding Business Activity. These roles become explicit when the transaction is used within a Business Transaction Activity within a Binary Collaboration. There is no need to make these roles more explicit such as buyer or seller. In particular some business transactions, for example "Cancel Purchase Order" may be used either way within the same binary collaboration definition.

There is always a Request document flow.

Whether a Response document is required is part of the definition of the Business Transaction. Some Business Transactions need this type of request and response, typically for the formation of a contract or agreement. Other Business Transactions are more like notifications, and have only a Request document flow.

An abstract superclass, Business Action, is the holder of attributes that are common to both Requesting Business Activity and Responding Business Activity.

There is a retryCount attribute on the Requesting Business Activity, which indicate the number of business level retries, in case there is a business protocol failure. The message id is different when retried and the receiving application will be responsible for detecting for business level duplicates using the unique id (like a purchase order number) in the business information.

### 7.4.3.2 Sample syntax

Here is a simple business transaction definition with just a requesting and responding document flow:

```
<BusinessTransaction name="Get Repair Order">
  <RequestingBusinessActivity name="Dealer"
    <DocumentEnvelope
      businessDocument="Get Repair Order"/>
    </RequestingBusinessActivity>
  <RespondingBusinessActivity name="OEM">
    <DocumentEnvelope isPositiveResponse="false"
      businessDocument="confirmBOD"/>
    <DocumentEnvelope isPositiveResponse="true"
      businessDocument="Show Repair Order" />
    </RespondingBusinessActivity>
  </BusinessTransaction>
```

Associated with each document flow can be one or more business signals acknowledging the document flow. These acknowledgment signals are not modeled explicitly but parameters associated with the transaction specify whether the signals are required or not.

The possible Document Flows and business signals within a Business Transaction are shown in Figure 7.

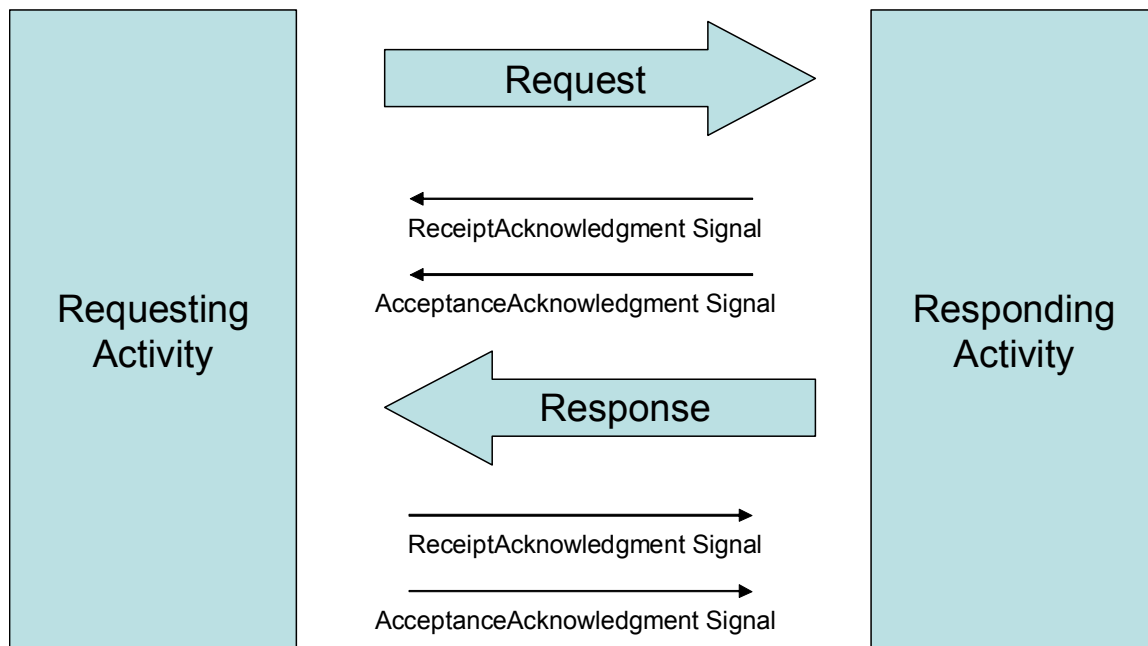


Figure 7: Possible document flows and signals and their sequence

These acknowledgment signals (a.k.a. Business Signals) are application level documents that 'signal' the current state of the business transaction.

Whether a receiptAcknowledgement and/or acceptanceAcknowledgement signal is required is part of the pattern specified for the Business Transaction. These business signals have specific business purposes, relating to the processing and management of documents and document envelopes *prior* to evaluation of their business terms, and are separate from lower protocol and transport signals.

The Receipt acknowledgement business signal, if used, signals that a message has been properly received by the ebXML Business Service Interface software component. The property *isIntelligentCheckRequired* allows partners to agree that a message should be confirmed by a Receipt acknowledgement only if it is also legible. Legible means that it has passed structure/ schema validity check. Both the proper receipt and, if evaluated, the legibility of a message are reviewed (and if present acknowledged) *prior* to the application of any business rules or evaluation

818 of the terms or guard expressions in the message's business documents  
819 or document envelope.

820 The Acceptance Acknowledgement business signal, if used, signals that  
821 the message received has been accepted for business processing by the  
822 receiving application, or a receiving business application proxy. This is  
823 the case if the contents of the message's business documents and  
824 document envelope have passed a business rule validity check. These  
825 business rules are not necessarily modeled as part of the collaboration.

826 Failure to send either signal, when required (by specifying a timeout value  
827 in `timeToAcknowledgeReceipt` or `timeToAcknowledgeAcceptance`), will  
828 result in the transaction being null and void, and therefore will prevent any  
829 "success" end state (protocol or business) that would have depended on  
830 receipt of a business document satisfying the associated `timeToPerform`.

831 Conversely, if all signals are positive and sent and received on time, the  
832 transaction will be successful from a protocol perspective.

833 The `isPositiveResponse` attribute of a `DocumentEnvelope` is not part of  
834 the business transaction protocol and therefore does not impact the  
835 protocol success or failure of a collaboration. If the `DocumentEnvelope`  
836 received as a response is specified with the `isPositiveResponse=false` (at  
837 design time) the business transaction will end in a business failure state.  
838 The choreography of the binary collaboration may use this information to  
839 execute corresponding transitions or stop the collaboration altogether.  
840 Note that this attribute is optional and some document envelope may  
841 neither be positive or negative (consider for instance the case of a partial  
842 acceptance on a purchase order, where only a few line items are refused,  
843 or a back order response). In this case, the business transaction activity  
844 is considered successful, again after it has reached a protocol success  
845 state.

846 The `isGuaranteedMessageDeliveryRequired` refers to the underlying  
847 messaging service used to implement the business transaction protocol.  
848 The business transaction protocol guarantees the processing of the  
849 business documents by the receiving application. However, to achieve  
850 this result, the business transaction protocol shall be implemented on top  
851 of a reliable messaging service that provides guaranteed message delivery  
852 at the transport level. In the case where the business transaction does not  
853 need to guarantee processing by the receiving application this condition  
854 can be relaxed and regular messaging services may be used.

855 It is important to note that we can only guarantee the complete  
856 synchronization of state if reliable messaging is used and if the business  
857 transaction is defined to use the request and response acceptance  
858 acknowledgement signals, which guarantee that the corresponding  
859 business documents were processed by the respective applications.

### 860 7.4.3.3 Sample syntax

861 Here is a slightly more complex transaction with two document flows and  
862 three business signals.

The request requires both receipt and acceptance acknowledgement, the response requires only receipt acknowledgement. "P2D" is a W3C Schema syntax adopted from the ISO 8601 standard and means Period=2 Days. P3D means Period=3 Days, P5D means Period=5 Days. These periods are all measured from original sending of request.

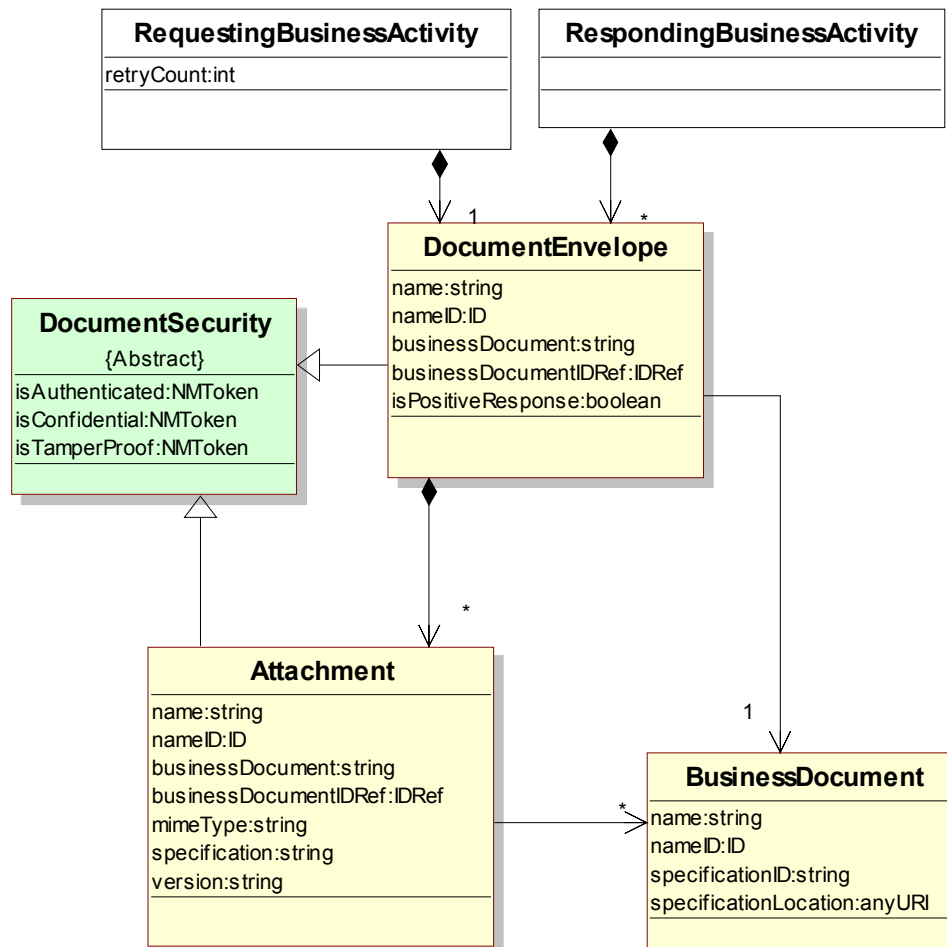
```

<BusinessTransaction
  name="Get Repair Order"
  nameID="122A3DD33"
  isGuaranteedDeliveryRequired="true">
  <RequestingBusinessActivity
    name="Dealer"
    nameID="122A3E833"
    isNonRepudiationReceiptRequired="false"
    isNonRepudiationRequired="false"
    timeToAcknowledgeAcceptance="P3D"
    timeToAcknowledgeReceipt="P2D">
    <DocumentEnvelope
      businessDocument="Get Repair Order"
      businessDocumentIDRef="122A3F613" />
  </RequestingBusinessActivity>
  <RespondingBusinessActivity
    name="OEM"
    nameID="122A3E863"
    isNonRepudiationReceiptRequired="false"
    isNonRepudiationRequired="false"
    timeToAcknowledgeReceipt="P1D">
    <DocumentEnvelope
      name="Reject Repair Order"
      isPositiveResponse="false"
      businessDocument="confirmBOD"
      businessDocumentIDRef="122A3F8E3" />
    <DocumentEnvelope
      name="Accept Repair Order"
      isPositiveResponse="true"
      businessDocument="Show Repair Order"
      businessDocumentIDRef="122A3F6C3" />
  </RespondingBusinessActivity>
</BusinessTransaction>

```

#### 7.4.3.4 Specifying Business Document flows

Request document flows and response document flows contain Business Documents that pertain to the Business Transaction. The model for this is shown in Figure 8. Business Documents have varying structures. Business signals, however always have the same structure, defined once and for all as part of the ebXML *Business Process Specification Schema*.



**Figure 8: UML Diagram of document flow**

A document flow is not modeled directly. Rather it is modeled indirectly as a Document Envelope sent by one role and received by the other. The Document Envelope is always associated with one Requesting Business Activity and one Responding Business Activity to model the flow.

Document Envelopes are named. There is always only one named Document Envelope for a Requesting Activity. There may be zero, one, or many mutually exclusive, named Document Envelopes for a Responding Activity. For example, the Response Document Envelopes for a purchase order transaction might be named `PurchaseOrderAcceptance`, `PurchaseOrderDenial`, and `PartialPurchaseOrderAcceptance`. In the actual execution of the purchase order transaction, however, only one of the defined possible responses will be sent.

The Document Envelope represents the flow of documents between the activities. Each Document Envelope carries exactly one primary Business Document.

930 A Document Envelope can optionally have one or more attachments, all  
931 related to the primary Business Document. The document and its  
932 attachments in essence form one transaction in the payload in the ebXML  
933 Message Service message structure.

#### 934 7.4.3.5 Sample syntax

935 This example shows a business transaction with one request and two  
936 possible responses, a success and a failure. The request has an  
937 attachment. All the Business Documents are fully qualified with the  
938 schema name.

```
939
940 <BusinessDocument
941     name="confirmBOD"
942     nameID="122A3F8E3"
943     specificationLocation="http://www.starstandard/xml/confirmBOD.
944 xsd"/>
945 <BusinessDocument
946     name="Get Repair Order"
947     nameID="122A3F613"
948     specificationLocation="http://www.starstandard/xml/ Get Repair
949 Order.xsd"/>
950 <BusinessDocument
951     name="Process Repair Order"
952     nameID="122A3FCB3"
953     specificationLocation="http://www.starstandard/xml/ Process
954 Repair Order.xsd"/>
955 <BusinessDocument
956     name="Show Repair Order"
957     nameID="122A3F6C3"
958     specificationLocation="http://www.starstandard/xml/ Show Repair
959 Order.xsd"/>
960 <BusinessDocument
961     name="Update Repair Order"
962     nameID="122A3FBE3"
963     specificationLocation="http://www.starstandard/xml/ Update
964 Repair Order.xsd"/>
965
966 <BusinessDocument
967     name="Repair Order Details"
968     nameID="122A3FBE4"
969     specificationLocation="http://www.starstandard/xml/ Repair Order
970 Details.xsd"/>
971
972
973 <BusinessTransaction name="Get Repair Order"
974     nameID="122A3DD33"
```

```
975             isGuaranteedDeliveryRequired="true">
976     <RequestingBusinessActivity
977       name="Dealer"
978       nameID="122A3E833"
979       isAuthorizationRequired="true"
980       isIntelligibleCheckRequired="true"
981       isNonRepudiationReceiptRequired="true"
982       isNonRepudiationRequired="true"
983       timeToAcknowledgeAcceptance=" PT30S"
984       timeToAcknowledgeReceipt=" PT10S">
985       <DocumentEnvelope
986         isAuthenticated="persistent"
987         isConfidential="persistent"
988         isTamperProof="persistent"
989         businessDocument="Get Repair Order"
990         businessDocumentIDRef="122A3F613C"/>
991     </RequestingBusinessActivity>
992
993     <RespondingBusinessActivity
994       name="OEM"
995       nameID="122A3E863"
996       isAuthorizationRequired="true"
997       isIntelligibleCheckRequired="true"
998       isNonRepudiationReceiptRequired="true"
999       isNonRepudiationRequired="true"
1000       timeToAcknowledgeReceipt="PT10S">
1001       <DocumentEnvelope
1002         isPositiveResponse="false"
1003         isAuthenticated="persistent"
1004         isConfidential="persistent"
1005         isTamperProof="persistent"
1006         businessDocument="confirmBOD"
1007         businessDocumentIDRef="122A3F8E3"/>
1008       <DocumentEnvelope
1009         isPositiveResponse="true"
1010         isAuthenticated="persistent"
1011         isConfidential="persistent"
1012         isTamperProof="persistent"
1013         businessDocument="Show Repair Order"
1014         businessDocumentIDRef="122A3F6C3">
1015         <Attachment
1016           name="DeliveryNotes"
1017           mimeType="XML"
1018           businessDocument="Repair Order Details"
1019           businessDocumentIDRef="122A3F8E4"
```



```

1020         isConfidential="none"
1021         isTamperProof="none"
1022         isAuthenticated="none">
1023     </Attachment>
1024 </DocumentEnvelope>
1025 </RespondingBusinessActivity>
1026 </BusinessTransaction>
1027
1028
1029

```

#### 7.4.4 Specify a Binary Collaboration

Figure 9 illustrates a binary collaboration.

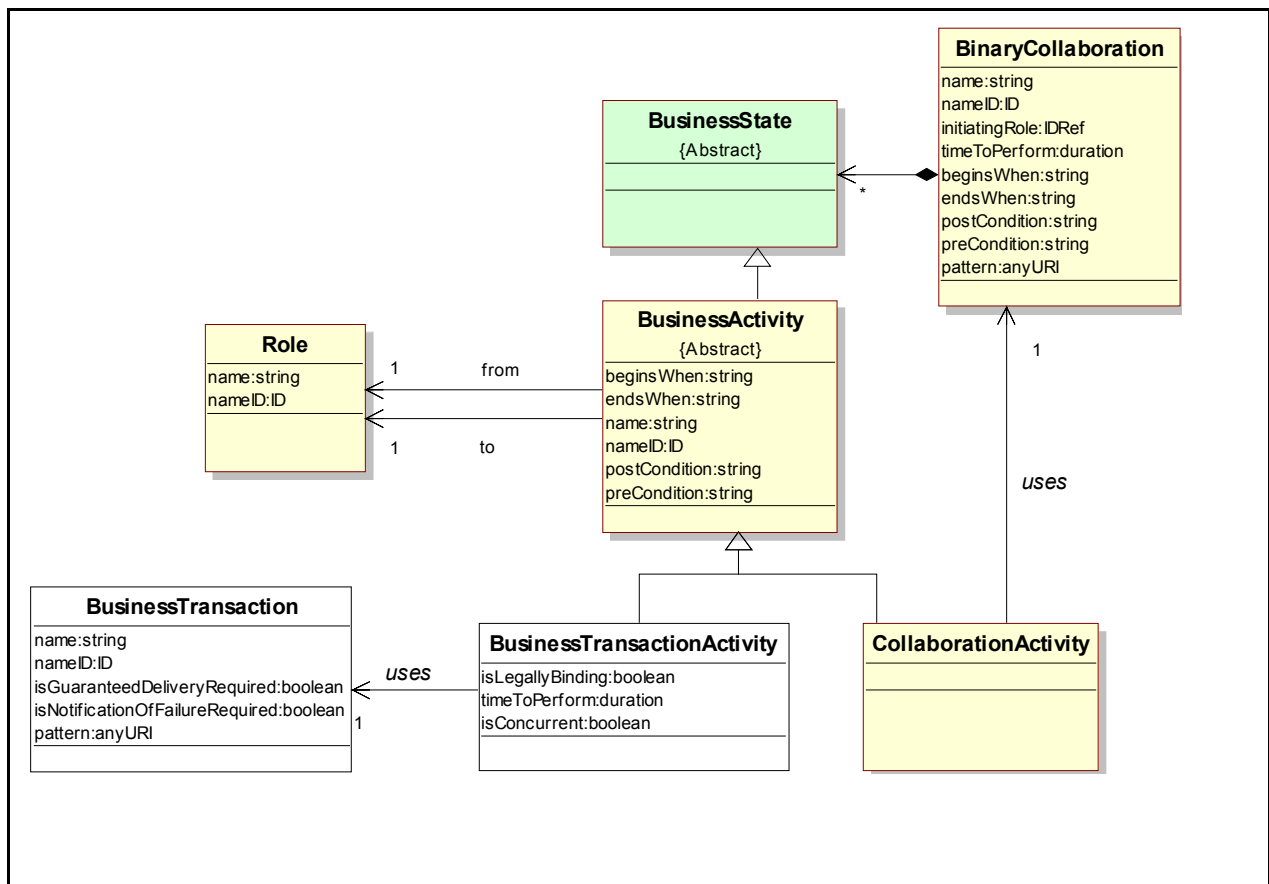


Figure 9: UML Diagram of a Binary Collaboration

#### 7.4.4.1 Key Semantics of a Binary Collaboration

A Binary Collaboration is always between two roles. One of the roles is initiating the collaboration. This is the role, which sends the first message. Note that in certain cases, the initiating role may not be known until runtime when the collaboration choreography starts with a fork, which involves business transaction activities, which can be initiated by the two partners.

A Binary Collaboration consists of one or more Business Activities. These Business Activities are always conducted **between** the two Roles of the Binary Collaboration. For each activity one of two roles is assigned to be the InitiatingRole (from) and the other to be the RespondingRole (to). This is irrespective of the Binary Collaboration roles.

A Business Activity can be either a Business Transaction Activity or a Collaboration Activity.

A Business Transaction Activity is the performance of a Business Transaction. Business Transactions are re-useable relative to Business Transaction Activity. The same Business Transaction can be performed by multiple Business Transaction Activities in different Binary Collaborations, or even by multiple Business Transaction Activities in the same Binary Collaboration, sometimes with opposite roles. For instance a "Cancel Purchase Order" Business Transaction could be modeled with two Business Transaction Activities, which can be performed by either party.

A Collaboration Activity is the performance of a Binary Collaboration, within another Binary Collaboration. Binary Collaborations are re-useable relative to Collaboration Activity. The same Binary Collaboration can be performed by multiple Collaboration Activities in different Binary Collaborations, or even by multiple Collaboration Activities in the same Binary Collaboration.

Business Transaction Activity and Collaboration Activity may define business rules with the beginsWhen, endsWhen, preCondition and postCondition attributes. These attributes do not have a specific syntax as part of this specification, so the current type is string. Because these expressions cannot be generally executed by any ebXML infrastructure they are just here as documentation and do not interfere with the choreography of the collaboration. In future specifications they will play a role along with transitions and pseudo-states. The semantics of beginsWhen and endsWhen indicate that the corresponding business activity needs to be started or ended as soon as the expression in the attribute value is true. PreConditions and postConditions indicate that the corresponding business activity may start only if the corresponding expressions are true.

When performing a Binary Collaboration within a Binary Collaboration there is an implicit relationship between the roles at the two levels. Assume that Binary Collaboration X is performing Binary Collaboration Y through Collaboration Activity Q. Binary Collaboration X has the following

roles: Customer and Vendor. In Collaboration Activity Q we assign Customer to be the initiator, and Vendor to be the responder. Binary Collaboration Y has the following roles: Buyer and Seller and a Business Transaction Activity where Buyer is the initiator and Seller the responder. We have now established a role relationship between the roles Customer and Buyer because they are both initiators in activities in the related performing and performed Binary Collaborations.

Since a Business Transaction is atomic in nature, the performing of a single Business Transaction through a Business Transaction Activity is also atomic in nature. If the desired semantic is not atomic, and then the task should be split over multiple transactions. For instance if it is desired to model several partial acceptances of a request, then the request should be modeled as one transaction within a binary collaboration and the partial acceptance(s) as separate transactions.

The CPA/CPD Specification allows that parties agree upon a Collaboration Protocol Agreement (CPA) in order to transact business. A CPA may associate itself with a specific Binary Collaboration. Thus, all Business Transactions performed between two parties should be referenced through Business Transaction Activities contained within a Binary Collaboration.

#### 7.4.4.2 Sample syntax

Here is a simple Binary Collaboration using one of the Business Transactions defined above:

```
<BinaryCollaboration
  name="Repair Order"
  nameID="122A38D93"
  initiatingRoleID="122A38DA3"
  timeToPerform="P3D">
  <Role
    name="Dealer"
    nameID="122A38DA3" />
  <Role
    name="OEM"
    nameID="122A38DA3" />
  <Start
    toBusinessState="Get Repair Order"
    toBusinessStateIDRef="122A39C23"
  <BusinessTransactionActivity
    name="Get Repair Order"
    nameID="122A39C23"
    businessTransaction="Get Repair Order"
    businessTransactionIDRef="122A3DD33"
    fromRole="Dealer"
    fromRoleIDRef="122A38DA3"
```

```

1130         toRole="OEM"
1131         toRoleIDRef="122A38DA3" isConcurrent="true"
1132         isLegallyBinding="false"
1133         timeToPerform="P1D"/>
1134     <Failure
1135         fromBusinessState="Get Repair Order"
1136         fromBusinessStateIDRef="122A39C23"
1137         conditionGuard="AnyProtocolFailure"/>
1138     <Success
1139         fromBusinessState="Get Repair Order"
1140         fromBusinessStateIDRef="122A39C23"
1141         conditionGuard="Success"/>
1142 </BinaryCollaboration>
1143
1144

```

## 7.4.5 Specify a MultiParty Collaboration

Figure 10 illustrates a multiparty collaboration

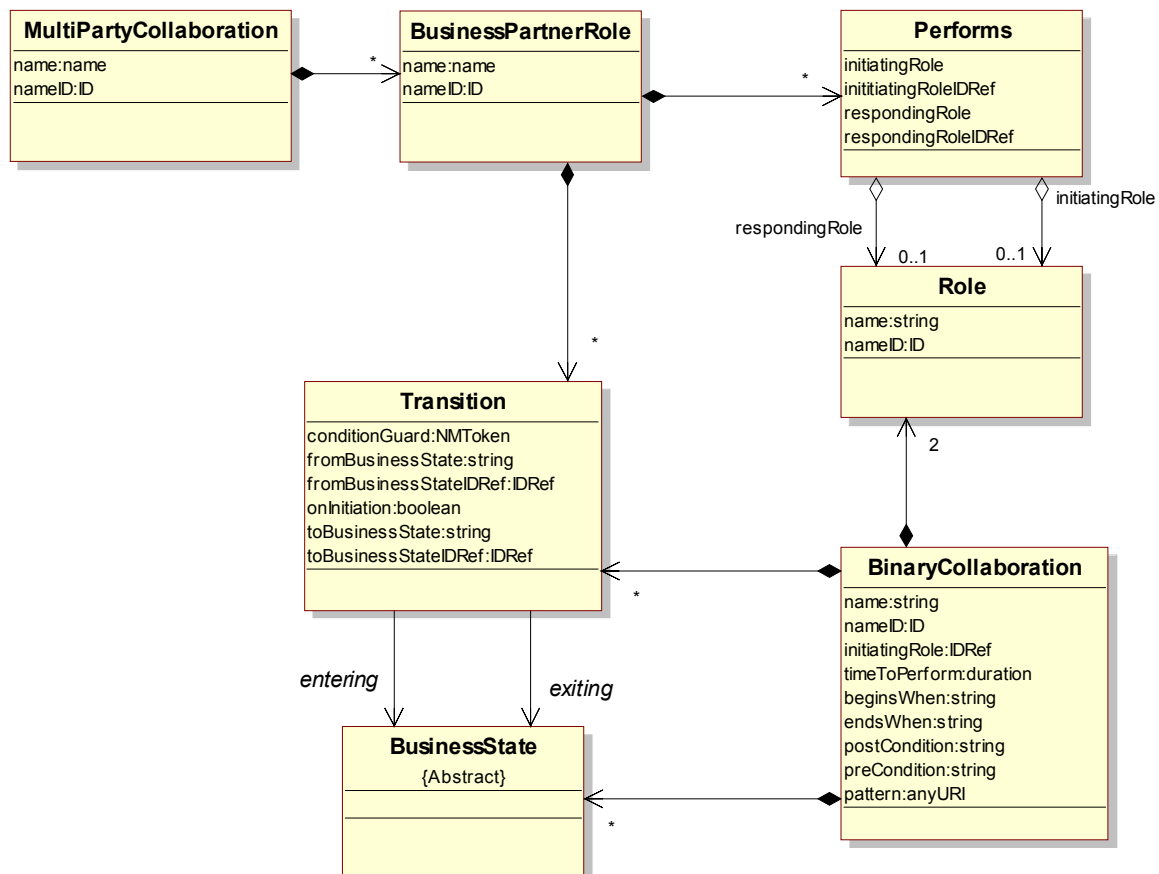


Figure 10: UML Diagram of a MultiParty Collaboration

1151

1152

### 1153 7.4.5.1 Key Semantics of a Multiparty Collaboration

1154 A Multiparty Collaboration is a synthesis of Binary Collaborations.

1155 A Multiparty Collaboration consists of a number of Business Partner  
1156 Roles.

1157 Each Business Partner Role performs one Role in one of the binary  
1158 collaborations, or perhaps one Role in each of several binary  
1159 collaborations. This is modeled by use of the Performs element.

1160 This 'Performs' linkage between a Business Partner Role and a Role is  
1161 the synthesis of Binary Collaborations into Multiparty Collaborations.  
1162 Implicitly the Multiparty Collaboration consists of all the Binary  
1163 Collaborations in which its Business Partner Roles play Authorized Roles.

1164 Each binary pair of trading partners may be subject to one or more  
1165 distinct CPAs.

1166 Within a Multiparty Collaboration, you may choreograph transitions  
1167 between Business Transaction Activities in different Binary  
1168 Collaborations, as described below.

1169

### 1170 7.4.5.2 Sample syntax

1171 Here is a simple Multiparty Collaboration which involves 3 parties  
1172 (Requester, Intermediary and Provider) performing the simple roles of  
1173 "sender" and "receiver". B is considered an intermediary. The same  
1174 binary collaborations is executed between each parties: one between the  
1175 Requester and the Intermediary and the other between the intermediary  
1176 and the Provider. In this case, the Intermediary plays both roles of the  
1177 Binary Collaboration.

```
1178 <MultiPartyCollaboration name="Send via intermediary">
1179   <BusinessPartnerRole name="Requester">
1180     <Performs role="sender"/>
1181   </BusinessPartnerRole>
1182   <BusinessPartnerRole name="Intermediary">
1183     <Performs role="receiver"/>
1184     <Performs role="sender"/>
1185   </BusinessPartnerRole>
1186   <BusinessPartnerRole name="Provider">
1187     <Performs role="receiver"/>
1188   </BusinessPartnerRole>
1189 </MultiPartyCollaboration>
```

1190

1191 Note that the role value links the corresponding Binary Collaboration  
1192 definition to this Multiparty Collaboration definition.

1193

## 7.4.6 Specify a Choreography

Figure 11 illustrates a choreography.

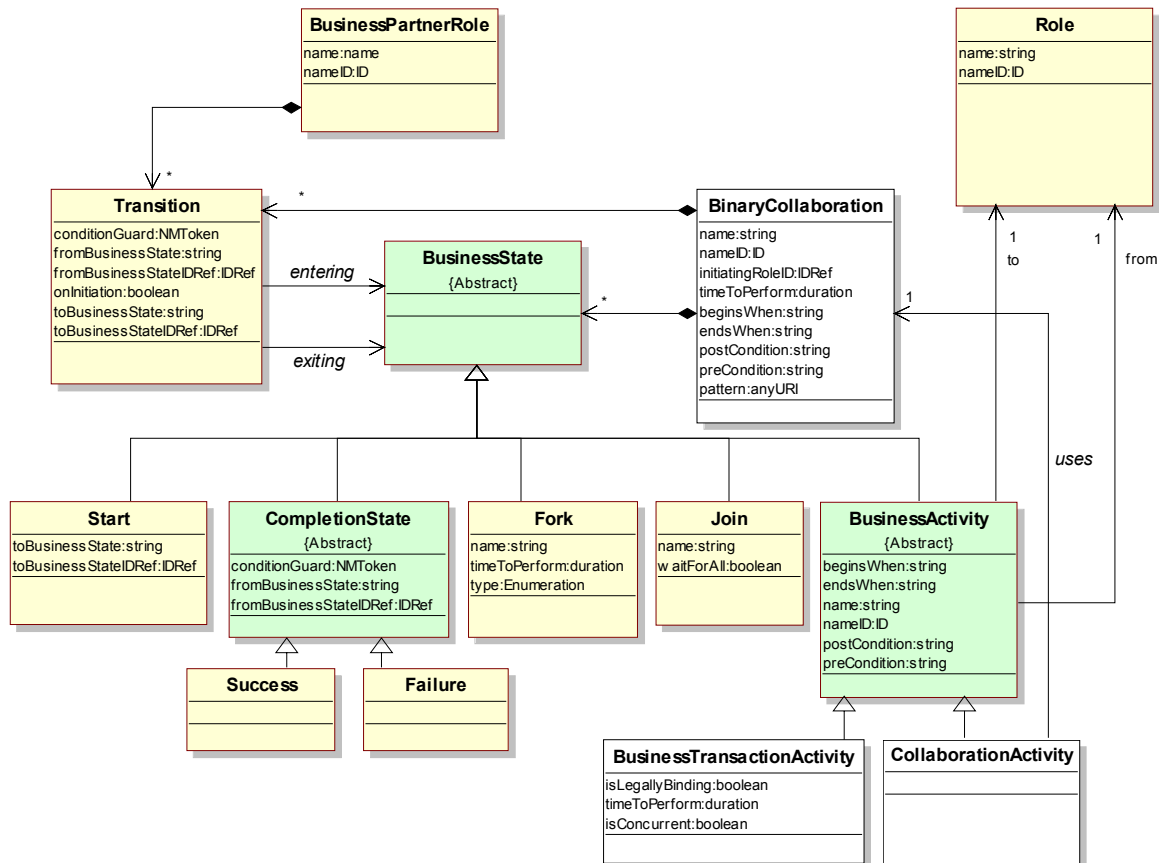


Figure 11: UML Diagram of a Choreography

### 7.4.6.1 Key Semantics of a Choreography

A Choreography is an ordering and sequencing of Business Activities within a Binary Collaboration.

The choreography is specified in terms of Business States, and transitions between those Business States.

A Business Activity is an abstract kind of Business State. Its two subtypes Business Transaction Activity and Collaboration Activity are concrete Business States. The purpose of a Choreography is to order and sequence Business Transaction Activity and/or Collaboration Activity within a Binary Collaboration, or across Binary Collaborations within a Multiparty Collaboration. The choreography specification and the business transaction protocol specifies unambiguously and at any point in

1213 time which message (DocumentEnvelope or Signal) are expected by any  
1214 of the parties. A message will either initiate a collaboration or advance its  
1215 state.

1216 There are a number of auxiliary kinds of Business States that facilitate the  
1217 choreographing of Business Activities. These include a Start state, a  
1218 Completion state (which comes in a Success and Failure flavor), a Fork  
1219 state and a Join state. These are all equivalent to diagramming artifacts  
1220 on a UML activity chart. A XOR value in the type attribute of a fork means  
1221 that only one transition coming out of the fork will execute. All the other  
1222 will become invalid. An ALL value will mean that any business activity  
1223 pointed to by a transition coming from the fork might be initiated. This  
1224 business activity may occur in parallel. Similarly, the waitForAll attribute of  
1225 the join will indicate that all transitions coming into the join shall be  
1226 executed in order to for the collaboration to reach the join pseudo-state.  
1227 The timeToPerform attribute on the Fork element may be used to specify  
1228 that the business activities between the Fork and the Join shall be  
1229 executed within the specified duration otherwise, the state of the  
1230 collaboration will automatically advance to the join. This feature is useful  
1231 in cases where the business activities are optional. For instance a Cancel  
1232 Purchase Order and Change Purchase Order business transaction  
1233 activity could be defined as part of a Fork/Join control block. However,  
1234 most often none of these activity would happen. The semantics of fork  
1235 and join are such that for instance a fork may be defined without a  
1236 corresponding join. In this case, the timeToPerform attribute shall not be  
1237 used. It must only be used in the case where all outgoing transitions from  
1238 the fork have incoming transitions to the join.

1239 Transitions can originate from Business Transaction Activity or  
1240 Collaboration Activity within a Binary Collaboration, or from Binary  
1241 Collaboration within a Multiparty Collaboration. Guards can gate  
1242 transitions. Guards refer to the status of the Business Transaction Activity  
1243 from which the transition originates. The guard values include:  
1244 ProtocolSuccess, AnyProtocolFailure, RequestReceiptFailure,  
1245 RequestAcceptanceFailure, ResponseReceiptFailure,  
1246 ResponseAcceptanceFailure, SignalTimeOut, ResponseTimeOut,  
1247 Failure, BusinessSuccess, BusinessFailure and Success. Transitions  
1248 may also have a condition expression element. A ConditionExpression  
1249 element has a language attribute, which specifies in which language the  
1250 predicate is written. We do not limit the type and number of languages a  
1251 BSI may support. However, for compliance, a BSI is required to support  
1252 at least the XPath language, as well as the DocumentEnvelopeLanguage.  
1253 An XPath expression may involve the content of any DocumentEnvelope  
1254 received prior to the transition within the scope of the current binary  
1255 collaboration instance. The DocumentEnvelopeLanguage is simply  
1256 defined as the name or ID of a document envelope.

1257 The Success and Failure elements represent an aggregation of a state  
1258 and a transition to this particular state. This transition like regular  
1259 transitions can be guarded by a conditionGuard. The conditionGuard can  
1260 be used to indicate that a binary collaboration ends in success or failure

based on the fact that the last business transaction activity response is a business document of a particular type, or based on the content of the response. It is important to note that the success or failure of the collaboration does not affect the success or failure of the individual business transaction activities, which compose the binary collaboration. In particular, the nature of the commitments is not changed when the collaboration ends in a specific state. The success or failure of a collaboration is rather an indication, which can be reported on, or acted upon to initiate other collaborations.

A Transition can also be used to create nested BusinessTransaction-Activities. A nested BusinessTransactionActivity is one where a first transition happens after the receipt of the request in the first transaction, and then the entire second transaction is performed before returning to the first transaction to send the response back to the original requestor. The flag 'onInitiation' in Transition is used for this purpose. Nested BusinessTransactionActivity are typically within a multiparty collaboration. In essence a Role in one Binary Collaboration receives a request, then turns around and becomes the requestor in other Binary Collaboration before coming back and sending the response in the first Binary Collaboration.

isConcurrent is a parameter that governs the flow of transactions. Unlike the security and timing parameters it does not govern the internal flow of a transaction, rather it determines whether multiple instances of that transaction type can be 'open' at the same time as part of the same business transaction activity. IsConcurrent is the parameter that governs this. It is at the business transaction activity level.

#### 7.4.6.2 Sample syntax

Here is the same Binary Collaboration as used before, with choreography added at the end. There is a transition between the two, a start and two possible outcomes of this collaboration, success and failure:

```
<BinaryCollaboration
  name="Repair Order"
  nameID="122A38D93"
  Role="233A38DA3"
  timeToPerform="P3D">
  <Role
    name="Dealer"
    nameID="122A38DA3" />
  <Role
    name="OEM"
    nameID="122A38DA3" />
  <Start
    toBusinessState=" Process Repair Order"
    toBusinessStateIDRef="122A39C23" />
```



```
1308         <BusinessTransactionActivity
1309             name="Process Repair Order"
1310             nameID="122A39C23"
1311             businessTransaction="Process Repair Order"
1312             businessTransactionIDRef="122A39C23"
1313             fromRole="Dealer"
1314             fromRoleIDRef="122A38DA3"
1315             toRole="OEM"
1316             toRoleIDRef="122A38DA3" isConcurrent="true"
1317             isLegallyBinding="false"
1318             timeToPerform="P1D"/>
1319         <BusinessTransactionActivity
1320             name="Update Repair Order"
1321             nameID="122A39CA3C3B0172"
1322             businessTransaction="Update Repair Order"
1323             businessTransactionIDRef="122A39CA3"
1324             fromRole="Dealer"
1325             fromRoleIDRef="122A38DA3"
1326             toRole="OEM"
1327             toRoleIDRef="122A38DA3" isConcurrent="true"
1328             isLegallyBinding="false"
1329             timeToPerform="P1D"/>
1330         <Success
1331             fromBusinessState="Process Repair Order"
1332             fromBusinessStateIDRef="122A39C23"
1333             conditionGuard="Success"/>
1334         <Success
1335             fromBusinessState="Update Repair Order"
1336             fromBusinessStateIDRef="122A39CA3"
1337             conditionGuard="Success"/>
1338         <Failure
1339             fromBusinessState="Process Repair Order"
1340             fromBusinessStateIDRef="122A39C23"
1341             conditionGuard="AnyProtocolFailure"/>
1342         <Failure
1343             fromBusinessState="Update Repair Order"
1344             fromBusinessStateIDRef="122A39CA3"
1345             conditionGuard="AnyProtocolFailure"/>
1346         <Transition
1347             onInitiation="false"
1348             fromBusinessState="Update Repair Order"
1349             fromBusinessStateIDRef="122A39CA3"
1350             toBusinessState="Process Repair Order"
1351             toBusinessStateIDRef="122A39C23"
1352             conditionGuard="Success">
1353             <ConditionExpression
1354                 expressionLanguage=
1355                     "DocumentEnvelopeLanguage"
1356                 conditionExpression=
1357                     "Accept Repair Order"/>
```

```

1358         </Transition>
1359     </BinaryCollaboration>
1360

```

1361        Optionally the transition with the condition expression could be expressed with an  
 1362        XPath predicate:

```

1363         <Transition
1364             onInitiation="false"
1365             fromBusinessState="Update Repair Order"
1366             fromBusinessStateIDRef="122A39CA3"
1367             toBusinessState="Process Repair Order"
1368             toBusinessStateIDRef="122A39C23"
1369             conditionGuard="Success">
1370             <ConditionExpression
1371                 expressionLanguage="XPath"
1372                 conditionExpression=
1373                     " //ConfirmBOD[@status='Reject'] "
1374             />
1375         </Transition>
1376

```

1377        Here is the same Multiparty Collaboration as defined before, but with a simple  
 1378        choreography (transition) across two Binary Collaborations. BTA is assumed to  
 1379        be the name of the Business Transaction Activity of the Binary Collaboration.  
 1380        Since the same Binary Collaboration is executed between all parties, the  
 1381        transition is from BTA (of the first collaboration) to BTA (of the second  
 1382        collaboration).

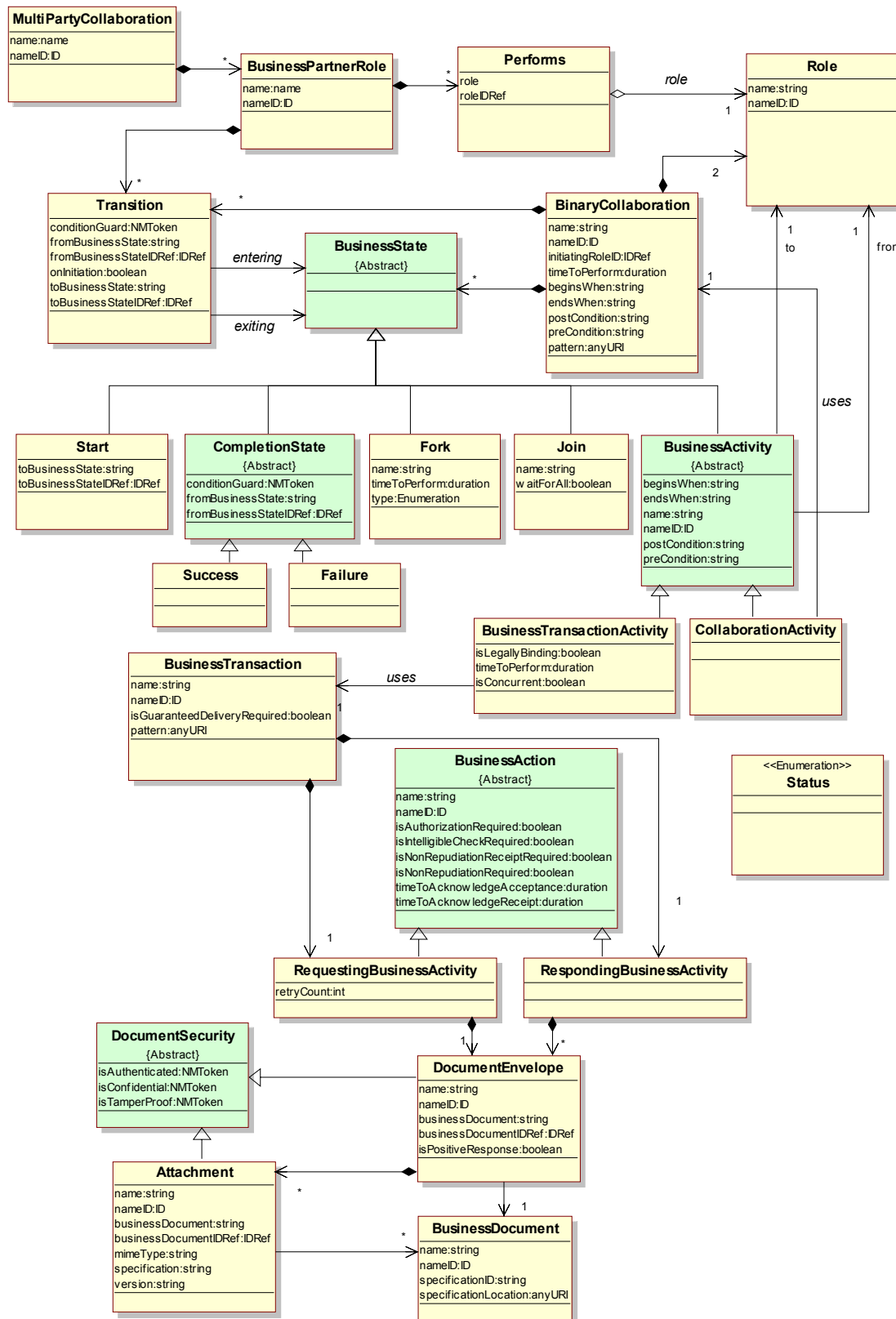
```

1383         <MultiPartyCollaboration name="Send via intermediary">
1384             <BusinessPartnerRole name="Requester">
1385                 <Performs role="sender" />
1386             </BusinessPartnerRole>
1387             <BusinessPartnerRole name="Intermediary">
1388                 <Performs role="receiver" />
1389                 <Performs role="sender" />
1390             <Transition
1391                 fromBusinessState="BTA"
1392                 toBusinessState="BTA" />
1393             </BusinessPartnerRole>
1394             <BusinessPartnerRole name="Provider">
1395                 <Performs role="receiver" />
1396             </BusinessPartnerRole>
1397         </MultiPartyCollaboration>
1398

```

## 1399 7.4.7 The whole model

1400  
 1401        Figure 12 shows the above semantics collectively as a UML class diagram. This  
 1402        diagram contains the whole UML version of the ebXML *Business Process*  
 1403        *Specification Schema*.



**Figure 12: Overall UML Specification Model of the ebXML Business Process Specification Schema**

## **7.5 Core Business Transaction Semantics**

The ebXML concept of a business transaction and the semantics behind it are central to predictable, enforceable commerce. It is expected that any Business Service Interface (BSI) will be capable of managing a transaction according to these semantics.

The ebXML Business Transaction semantics allows you to specify electronic commerce transactions that provide

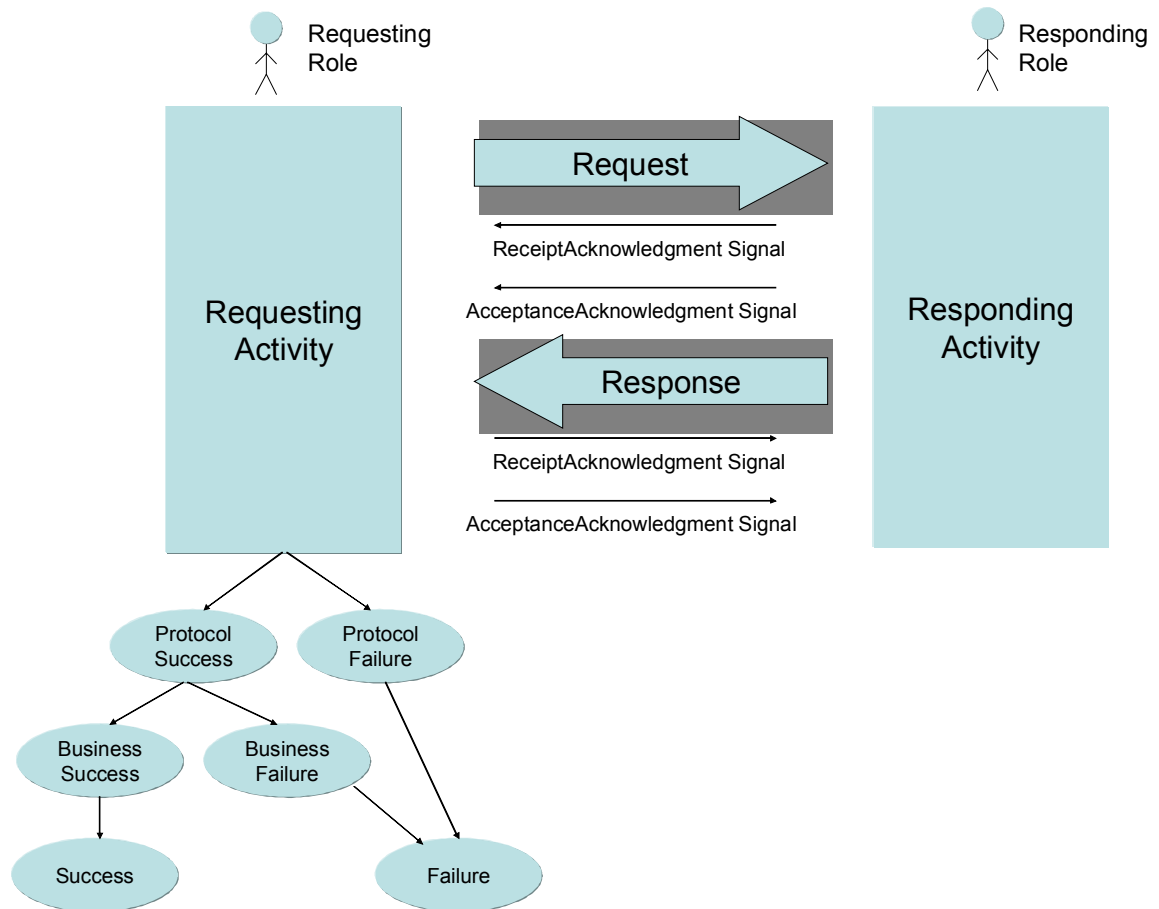
- Interaction Predictability, i.e. have clear roles, clear transaction scope, clear time bounds, clear business information semantics, clear determination of success or failure. Each party can compute without ambiguity and the status of a transaction independently.
- Ability to create Legally Binding Contracts, i.e. the ability to specify that Business Transactions may be agreed to bind the parties.
- Nonrepudiation, i.e. may specify the keeping of artifacts to aid in legal enforceability.
- Authorization Security, i.e. may be specified to require authorization of parties performing roles.
- Document Security, i.e. may be specified to be authorized, authenticated, confidential, tamperproof.
- Reliability, i.e. the ability to specify reliable delivery of Business Documents and signals.
- Run time Business Transaction Semantics, i.e. the rules and configuration parameters required for Business Service Interface software to predictably and deterministically execute ebXML Business Transactions.

Each of the above characteristics of ebXML Business Transaction semantics is discussed in detail below.

### **7.5.1 Interaction Predictability**

All Business Transactions follow a very precisely prescribed flow, or a precisely defined subset there-of. The following is an overall illustration of this flow. It can be thought of as the state machine across the two business partners. The N090R10 chapter on the UMM metamodel has a detail state chart for each of the business partners.

The goal of the Business Transaction Protocol is to synchronize the business state between two parties. As few resources can be shared between company boundaries, we must use such protocol to achieve the business state synchronization as recorded by each party enterprise systems.



**Figure 13: Schematic of core Business Transaction semantics.**

In the ebXML model the business transaction always has the following semantics.

1. The Business Transaction is a unit of work. All of the interactions in a business transaction must succeed or each party must not change their state.
2. A Business Transaction is conducted between two business partners playing opposite roles in the transaction. These roles are always the Requesting Role and the Responding Role.
3. A Business Transaction definition specifies exactly when the Requesting Activity is in control, when the Responding Activity is in control, and when control transitions from one to the other. In all Business Transactions control starts at the Requesting Activity, then transitions to the Responding Activity, and then returns to the Requesting Activity.
4. A Business Transaction always starts with a request sent out by the requesting activity.
5. The request serves to transition control to the responding role.

- 1465 6. After the receipt of the Request document flow, the responding activity  
 1466 may send a receiptAcknowledgement signal and/or an  
 1467 acceptanceAcknowledgement signal to the requesting role.
- 1468 7. The responding role then enters a responding activity. During or upon  
 1469 completion of the responding activity zero or one response is sent.
- 1470 8. Control will be returned back to the requesting activity if either a  
 1471 receiptAcknowledgement and/or acceptanceAcknowledgement and/or a  
 1472 response is specified as required. A receiptAcknowledgement (if required)  
 1473 must always occur before an acceptanceAcknowledgement (if required),  
 1474 and an acceptanceAcknowledgement must always occur before a  
 1475 response (if required). Control is returned to the requesting activity based  
 1476 on the last required of these three (if any). If none required, control stays  
 1477 with the responding activity.
- 1478 9. All business transactions succeed or fail. Success or failure depends on:
- 1479 a. The successful transmission of the request, the response and/or  
 1480 receipt and acceptance signals
- 1481 b. The occurrence of time-outs
- 1482 c. The occurrence of exceptions, as indicated by a negative receipt  
 1483 or acceptance signals
- 1484 d. The computation of business failure or success by detecting if the  
 1485 response document was specified – at design time – with  
 1486 isPositiveResponse=false.
- 1487 10. Both parties can compute the success or failure of the transaction if  
 1488 reliable messaging as well as request and response acceptance  
 1489 acknowledgement signals is used. Once success or failure is thus  
 1490 established, the Business Transaction is considered closed with respect  
 1491 to both parties.
- 1492 11. Upon receipt of a response the requesting activity may send a  
 1493 receiptAcknowledgement and/or acceptanceAcknowledgement signal  
 1494 back to the responding role. This operation does not pass control back to  
 1495 the responding activity. If the requesting party send the signals after the  
 1496 timeout has occurred, the transaction is considered null and void.
- 1497 12. Upon identifying a time-out or exception in the processing of a Business  
 1498 Transaction each party will close the transaction and end in a protocol  
 1499 failure state.

1500

### 1501 7.5.1.1 Transaction Interaction Patterns

1502

1503 The business transaction specification will specify whether a requesting  
 1504 document requires a responding substantive document in order to achieve a  
 1505 "success" end state. In addition, the transaction may specify a proper nonzero  
 1506 time duration for timeToPerform, imposing a deadline for the substantive  
 1507 response.

1508 Furthermore, the specification of a business transaction may indicate, for the  
1509 request whether receiptAcknowledgement and/or acceptanceAcknowledgement  
1510 are required, and for the response whether receiptAcknowledgement and/or  
1511 acceptanceAcknowledgement are required.

1512 The way to specify that a receiptAcknowledgement is required is to set the  
1513 parameter timeToAcknowledgeReceipt to any proper time duration other than  
1514 zero. If this parameter has been set to a proper nonzero time duration, optionally  
1515 either or both of the isIntelligibleCheckRequired and  
1516 isNonrepudiationOfReceiptRequired parameters may also be set to 'Yes'.

1517 The way to specify that a acceptanceAcknowledgement is required is to set the  
1518 parameter timeToAcknowledgeAcceptance to any proper time duration other  
1519 than zero.

1520 So these two acknowledgement related parameters double as Boolean flags for  
1521 whether the signal is required as part of the transaction, and as values for time-  
1522 out of the transaction if the signal is not received.

1523 The specification of a business transaction may require each one of these  
1524 signals independently of whether the other is required. If one is not required, it is  
1525 actually not allowed. Therefore there is a finite set of combinations. The UMM  
1526 supplies an illustrative set of patterns representing those combinations, for  
1527 potential re-use.

1528

## 1529 7.5.2 Creating legally binding contracts

1530

1531 Trading partners may wish to indicate that a Business Transaction performed as  
1532 part of an ebXML arrangement is, or is not, intended to be binding. A  
1533 declaration of intent to be bound is a key element in establishing the legal  
1534 equivalence of an electronic message to an enforceable signed physical writing.  
1535 Parties may create explicit evidence of that intent by (1) adopting the ebXML  
1536 Business Process Specification Schema standard and (2) manipulating the  
1537 parameter ("isLegallyBinding") designated by the standard to indicate that intent.

1538 In some early electronic applications, trading partners have simply used the  
1539 presence, or absence, of an electronic signature (such as under the XML-DSIG  
1540 standard) to indicate that intent. However, documents which rely solely on the  
1541 presence of a signature may or may not be correctly interpreted, if there is  
1542 semantic content indicating that a so-called contract is a draft, or nonbinding, or  
1543 the like.

1544 In ebXML, the presence or absence of an electronic signature cannot indicate by  
1545 itself determine legally binding assent, because XML-DSIG signatures are  
1546 reserved for other uses as an assurance of sender identity and message  
1547 integrity.

1548 isLegallyBinding is a parameter at the BusinessTransactionActivity level, which  
1549 means that the performing of a BusinessTransaction within a Binary  
1550 Collaboration is either specified as legally binding or not.

1551

1552 When operating under this standard, parties form binding agreements by  
1553 exchanging binding messages that agree to terms (e.g., offer and acceptance).  
1554 The "isLegallyBinding" parameter is Boolean, and its default value is "true."  
1555 Under this standard, the exclusive manner for indicating that a Business Activity  
1556 is not intended to be binding is to include a "false" value for the "isLegallyBinding"  
1557 parameter for the transaction activity. As in EDI, the ebXML standard assumes  
1558 that Business Transactions are intended by the trading parties to be binding  
1559 unless otherwise indicated.

1560 As a non-normative matter, parties may wish to conduct nonbinding transactions  
1561 for a variety of reasons, including testing, and the exchange of proposed offers  
1562 and counteroffers on a non-committal basis so as to discover a possible agreed  
1563 set of terms. When using tangible signed documents, parties often do so by  
1564 withholding a manual signature, or using a "DRAFT" stamp. In ebXML, trading  
1565 partners may indicate that result by use of the "isLegallyBinding" parameter. See  
1566 the illustrative Simple Negotiation Pattern set forth in the ebXML E-Commerce  
1567 Patterns.

### 1568 7.5.3 Non-Repudiation

1569 Trading partners may wish to conduct legally enforceable business transactions  
1570 over ebXML. A party may elect to use non-repudiation protocols in order to  
1571 generate documentation that would assist in the enforcement of the contractual  
1572 obligation in court, in the case that the counterparty later attempts to repudiate its  
1573 ebXML Business Documents and messages.

1574 Repudiation generally refers to the ability of a trading partner to argue at a later  
1575 time, based on the persistent artifacts of a transaction, that it did not agree to the  
1576 transaction. That argument might be based on assertions that a replying  
1577 document was not sent, or was not sent by the proper party, or was incorrectly  
1578 interpreted (under the applicable standard or the trading partners' business rules)  
1579 as forming agreement.

1580 There are two kinds of non-repudiation protocol available under this document.  
1581 Each protocol provides the user with some degree of additional evidentiary  
1582 assurance by creating or requesting additional artifacts that would assist in a  
1583 later dispute over repudiation issues. Neither is a dispositive absolute assurance.  
1584 As in the paper world, trading partners are always free to invent colorful new  
1585 arguments that an apparently-enforceable statement should be ignored. These  
1586 parameters simply offer some opportunities to make that more difficult.

1587 One imposes a duty on each party to save copies of all Business Documents and  
1588 Document Envelopes comprising the transaction in the form they were  
1589 received(e.g. save in encrypted form if they were received in encrypted form) ,  
1590 each on their own side, i.e., requestor saves his request, responder saves his  
1591 response. This is the isNonRepudiationRequired parameter in the requesting or  
1592 responding activity. It is logically equivalent to a request that the other trading  
1593 partner maintain an audit trail. However, failure to comply with that request is not  
1594 necessarily computationally detectable at run time, nor would it override the  
1595 determination of a "success" or "failure" end state.



1596 The other requires the receiver of a business document to send a signed receipt,  
1597 which the original sender saves. This is the `isNonRepudiationOfReceiptRequired`  
1598 parameter in the requesting and responding business activity.

1599 `NonRepudiationOfReceipt` is tied to the `ReceiptAcknowledgement`, in that it  
1600 requires the latter to be digitally signed. So `NonRepudiationOfReceipt` is  
1601 meaningless if `ReceiptAcknowledgement` is not required. Failure to comply with  
1602 `NonRepudiationOfReceipt` would be computationally detectable at run time, and  
1603 would override the determination of a "failure" end state. If a  
1604 `timeToAcknowledgeReceipt` is imposed on a requesting message, and  
1605 `NonRepudiationOfReceipt` is true, only a digitally signed receipt will satisfy the  
1606 imposed timeout deadline. Thus, a failure to send a *signed* receipt within  
1607 `timeToAcknowledgeReceipt`, would make the transaction null and void.

#### 1608 7.5.4 Authorization security 1609

1610 Each request or response may be sent by a variety of individuals,  
1611 representatives or automated systems associated with a business partner.  
1612 There may be cases where trading partners have more than one ebXML-capable  
1613 business service interface, representing different levels of authority. In such a  
1614 case, the parties may establish rules regarding which interfaces or authors may  
1615 be confidently relied upon as speaking for the enterprise.

1616 In order to invoke those rules, a party may specify `IsAuthorizationRequired` on a  
1617 requesting or and responding activity accordingly, with the result that [the activity]  
1618 will only be processed as valid if the party interpreting it successfully matches the  
1619 stated identity of the activity's [Authorized Role] to a list of allowed values  
1620 previously supplied by that party.

1621 `IsAuthorizationRequired` is specified on the requesting and responding activity  
1622 accordingly.

1623

#### 1624 7.5.5 Document security

1625 The value of *isConfidential*, *isTamperProof*, *isAuthenticated* at the Document  
1626 Envelope always applies to the primary Business Document. It also applies to  
1627 each of the attachments unless specifically overridden at the Attachment level.  
1628 These parameters can have four possible values: none, transient, persistent,  
1629 transient-and-persistent.

1630 Transient authentication is provided by the communications channel used to  
1631 transport the *Message*. The specific method will be determined by the  
1632 communications protocol used.

1633 Persistent authentication means the Business Document signer's identity shall be  
1634 verified at the receiving application level.

1635 Transient confidentiality is provided by a secure network protocol, such as SSL  
1636 as the document is transferred between two adjacent MSH nodes.

1637 Persistent confidentiality is intended to preserve the confidentiality of the  
1638 message such that only the intended party (application) can see it. The message

1639 shall remain in encrypted form after it is delivered to the MSH node and will be  
1640 decrypted only by the authorized application. S/MIME can be used to provide that  
1641 functionality, independent of the transient confidentiality.

1642 Transient isTamperPoof is the ability to detect if the information has been  
1643 tampered with during transfer between two adjacent MSH nodes.

1644 Persistent isTamperDetectable is the ability to detect if the information has been  
1645 tampered with after it has been received by MSH, between the MSH and the  
1646 application.

1647

## 1648 7.5.6 Reliability

1649 This parameter IsGuaranteedDeliveryRequired at the Business Transaction level  
1650 states whether guaranteed delivery of the transaction's Business Documents is  
1651 required.

1652 This is a declaration that trading partners must employ only a delivery channel  
1653 that provides a delivery guarantee, to send Business Documents in the relevant  
1654 transaction.

1655

## 1656 7.5.7 Parameters required for CPP/CPA

1657

1658 The ebXML *Business Process Specification Schema* provides parameters that  
1659 can be used to specify certain levels of security and reliability. The ebXML  
1660 *Business Process Specification Schema* provides these parameters in general  
1661 business terms.

1662 These parameters are generic requirements for the business process, but for  
1663 ebXML implementations, these parameters are specifically used to instruct the  
1664 CPP and CPA to require BSI and/or delivery channel capabilities to achieve the  
1665 specified service levels.

1666 The CPP and CPA translate these into parameters of two kinds.

1667 One kind of parameter determines the selection of certain security and reliability  
1668 parameters applicable to the transport method and techniques used by the  
1669 delivery channel. Document security, and Reliability above, are determinators of  
1670 delivery channel selection.

1671 The other kind of parameter determines the selection of certain service levels or  
1672 capabilities of the BSI itself, in order for it to support the run time Business  
1673 Transaction semantics as listed below.

## 1674 7.6 Run time Business Transaction semantics

1675 The ebXML concept of a business transaction and the semantics behind it are  
1676 central to predictable, enforceable commerce. It is expected that any Business  
1677 Service Interface (BSI) will be capable of managing a transaction according to  
1678 these semantics.

1679 Therefore, the Business Service Interface (BSI), or any software that implements  
 1680 one role in an ebXML collaboration needs at minimum to be able to support the  
 1681 following transaction semantics:

- 1682 1. Detection of the opening of a transaction
- 1683 2. Detection of transfer of control
- 1684 3. Detection of successful completion of a transaction
  - 1685 a. Application of business rules expressed as isPositiveResponse
  - 1686 and transition conditionGuard for determination of success
- 1687 4. Detection of failed completion of a transaction
  - 1688 a. Detection of time-outs
  - 1689 b. Detection of protocol exceptions
  - 1690 c. Calculation of the received response to identify if it was specified
  - 1691 with isPositiveResponse = false

1692 ebXML does not specify how these transaction semantics are implemented but it  
 1693 is assumed that any Business Service Interface (BSI) will be able to support  
 1694 these basic transaction semantics at runtime. If either party cannot provide full  
 1695 support, then the requirements may be relaxed as overrides in the CPP/CPA.

1696 The following sections discuss the two causes of failure: Time-outs and  
 1697 Exceptions. When either one happens, it is the responsibility of the two roles to  
 1698 exit the transaction. It is also expected that the corresponding collaboration will  
 1699 be designed (and choreographed) to execute the appropriate compensating  
 1700 transactions if needed. The responsibilities of the two roles differ slightly and are  
 1701 described in each of the sections below. Generally, if a failure happens at either  
 1702 the responding or requesting role, they will send an exception signal to the other  
 1703 role, and both parties will exit the current transaction.

1704

1705

## 1706 7.6.1 Timeouts

1707

1708 Since all business transactions must have a distinct time boundary, there are time-  
 1709 out parameters associated with the response, and each of the acknowledgement  
 1710 signals. If the time-out occurs before the corresponding response or signal arrives,  
 1711 the transaction is null and void.

1712

1713 Here are the time-out parameters relative to the three response types:

1714

Response required	Parameter Name	Meaning of timeout

Receipt acknowledgement	timeToAcknowledgeReceipt	The time a responding or requesting role has to acknowledge receipt of a business document.
Acceptance Acknowledgement (Non-substantive)	timeToAcknowledgeAcceptance	The time a responding or requesting role has to non-substantively acknowledge business acceptance of a business document.
Substantive Response	TimeToPerform	The time a responding role has to substantively acknowledge business acceptance of a business document.

1715

1716

1717

1718

A time-out parameter must be specified whenever a requesting partner expects one or more responses to a business document request. A requesting partner must not remain in an infinite wait state.

1719

1720

1721

1722

The time-out value for each of the time-out parameters is absolute i.e. not relative to each other. All timers start when the initial requesting business document is sent. The timer values must comply with the well-formedness rules for timer values.

1723

1724

1725

A BSI needs to comply with the above parameters to detect the appropriate time outs. To preserve the atomic semantics of the Business Transaction, the requesting and responding roles take different action based on time outs.

1726

1727

A responding partner simply terminates if a timeout is thrown. This prevents responding business transactions from hanging indefinitely.

1728

1729

1730

1731

When the time to perform an activity equals the time to acknowledge receipt or the time to acknowledge business acceptance then the highest priority time out exception must be used when the originator provides a reason for revoking their original business document offer. The time to perform exception is lower priority

1732 than both the time to acknowledge receipt and the time to acknowledge business  
1733 acceptance.

1734

## 1735 7.6.2 Protocol Exceptions

1736

1737 In addition to timeouts, the Business Transaction protocol provides a series of  
1738 protocol exception which indicate whether the business processing of the  
1739 transaction went wrong at either the responding or the requesting role.

### 1740 7.6.2.1 Receipt Acknowledgement Exception

1741

1742 A *Receipt Exception* signals an error condition in the management of a business  
1743 transaction. This business signal is returned to the initiating activity that  
1744 originated the request. This exception must terminate the business transaction.  
1745 These errors deal with the mechanisms of message exchange such as  
1746 verification, validation, authentication and authorization and will occur up to  
1747 message acceptance. Typically the rules and constraints applied to the message  
1748 will have only dealt with the well-formness of the message.

1749 A receipt exception terminates the business transaction. The following are receipt  
1750 exceptions:

- 1751 • Syntax exceptions. There is invalid punctuation, vocabulary or  
1752 grammar in the business document or business signal.
- 1753 • Authorization exceptions. Roles are not authorized to participate in  
1754 the business transaction.
- 1755 • Signature exceptions. Business documents are not signed for  
1756 non-repudiation when required.
- 1757 • Sequence exceptions. The order or type of a business document  
1758 or business signal is incorrect.

1759 A receipt exception typically means that the current message could not be  
1760 passed to an application for processing.

### 1761 7.6.2.2 Acceptance Acknowledgement Exceptions

1762

1763 An Acceptance Exception signals an error condition in a business activity. This  
1764 business signal is returned to the initiating role that originated the request. This  
1765 exception must terminate the *business transaction*. These errors deal with the  
1766 mechanisms that process the *business transaction* and will occur after message  
1767 verification. Typically the rules and constraints applied to the message will deal  
1768 with the semantics of message elements and the validity of the request itself. The  
1769 content is not valid with respect to a responding role's business rules.

1770 An Acceptance Exception terminates the business transaction. The following are  
1771 business protocol exceptions:

- 1772                   • Business exception. The business rules of the responding activity  
1773                   are violated. The application refused to process the incoming  
1774                   business document.
- 1775                   • Performance exceptions. The requested business action cannot  
1776                   be performed. The application may not be available.
- 1777           Typically, an Acceptance Exception means that the processing application  
1778           (usually unknown to the other party) received the corresponding business  
1779           document but was unable to process them.
- 1780           A Business Transaction is defined in very atomic and deterministic terms. It  
1781           always is initiated by the requesting role, and will always conclude at the  
1782           requesting role. Upon receipt of the required response and/or signals, or time-out  
1783           of same, the requesting role can unambiguously determine the success or failure  
1784           of the Business Transaction. A responding role that encounters an Acceptance  
1785           Exception signals the exception back to the requesting role and then terminates  
1786           the business transaction.
- 1787           Conversely, a requesting role that encounters an Acceptance Exception signals  
1788           the exception back to the responding role and terminates the transaction
- 1789           A BSI needs to comply specifically with the following parameters to produce the  
1790           associated special exceptions. The requesting and responding roles take  
1791           different action as per below.
- 1792                   ***IsAuthorizationRequired***
- 1793                   If a partner role needs authorization to request a business action  
1794                   or to respond to a business action then the sending partner role  
1795                   must sign the business document exchanged and the receiving  
1796                   partner role must validate this business control and approve the  
1797                   authorizer. A responding partner must signal an authorization  
1798                   exception (receipt exception) if the requesting partner role is not  
1799                   authorized to perform the business activity. A sending partner  
1800                   must send notification of failed authorization if a requesting  
1801                   partner is not authorized to perform the responding business  
1802                   activity.
- 1803                   ***IsNonRepudiationRequired***
- 1804                   If non-repudiation of origin and content is required then the  
1805                   business activity must store the business document in its original  
1806                   form for the duration mutually agreed to in a trading partner  
1807                   agreement. A responding partner must signal a receipt exception  
1808                   if the sending partner role has not properly delivered their  
1809                   business document. Similarly, a requesting partner must send  
1810                   receipt exception if a responding partner has not properly  
1811                   delivered their business document.
- 1812                   ***isNonRepudiationOfReceiptRequired.***
- 1813                   Both partners agree to mutually verify receipt of a requesting  
1814                   business document and that the receipt must be non-repudiable.  
1815                   A requesting partner must initiate a notification of failure business

1816 transaction business (possibly revoking a contractual offer) if a  
1817 responding partner has not properly delivered signed their receipt.  
1818 For a further discussion of nonrepudiation of receipt, see also the  
1819 ebXML E-Commerce and Simple Negotiation Patterns.

1820  
1821 Non-repudiation of receipt provides the data for the following audit  
1822 controls.

1823 **Verify responding role identity** (authenticate) – Verify the  
1824 identity of the responding role (individual or organization) that  
1825 received the requesting business document.

1826 **Verify content integrity** – Verify the integrity of the original  
1827 content of the business document request.

1828 ***isPositiveResponse***

1829 An expression whose evaluation results in TRUE or FALSE. If  
1830 TRUE this DocumentEnvelope is intended as a positive response  
1831 to the request. If isPositiveResponse = FALSE, the business  
1832 transaction activity ends in business failure mode. The value for  
1833 this parameter supplied for a DocumentEnvelope is an assertion  
1834 by the sender of the DocumentEnvelope regarding its intent for  
1835 the transaction to which it relates, but does not bind the recipient,  
1836 or override the computation of transactional success or failure.

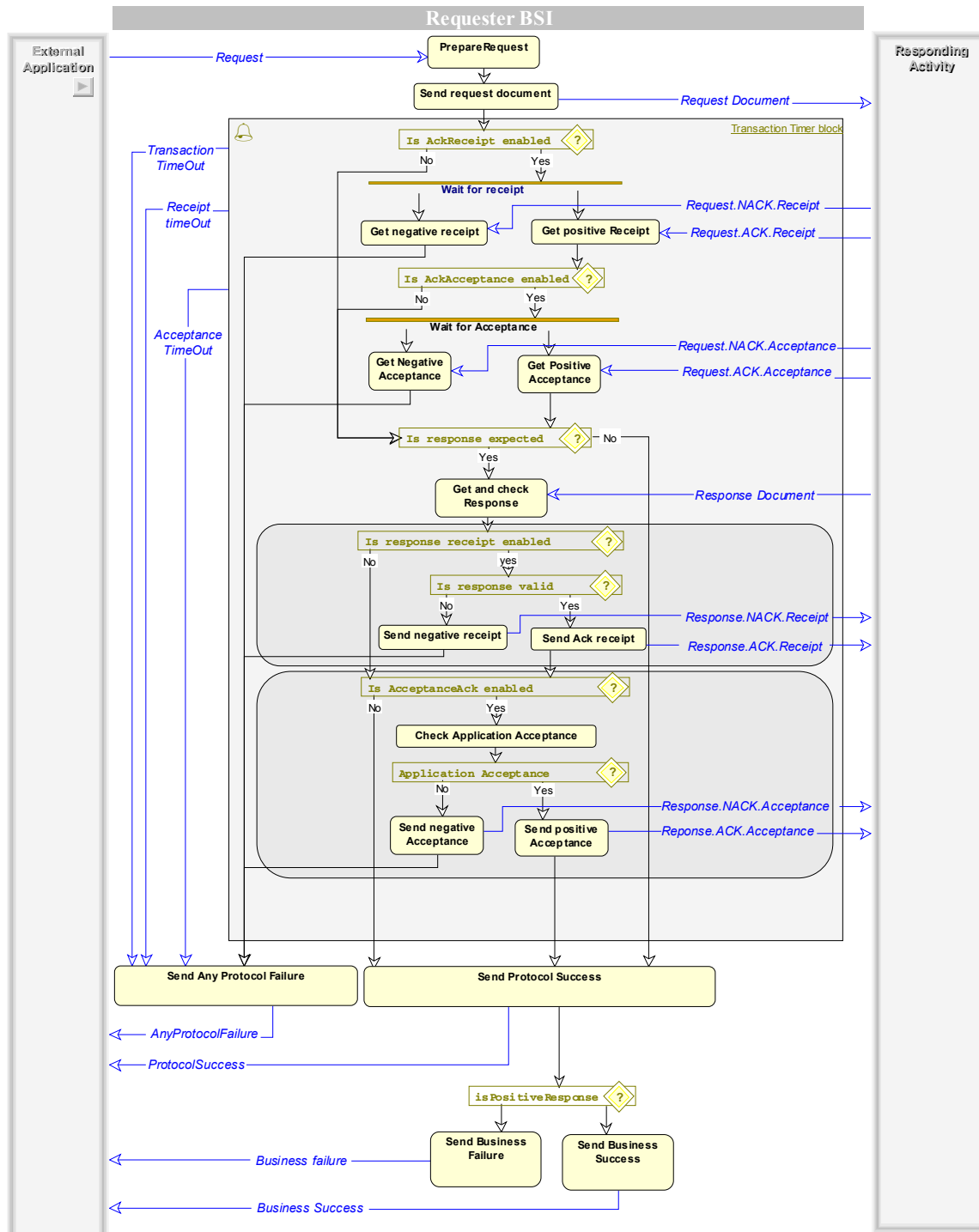
1837

### 1838 7.6.3 Computation of the status of a Business Transaction Activity

1839 The following figure represent the computation of the success or failure of a business  
1840 transaction activity based on the different possible scenarios.

1841

1842



**Fig 14. Computation of the Status of a Business Transaction Activity**

The values of the enumeration of the state of a business transaction of the conditionGuard on a transition are:

- ProtocolSuccess
- AnyProtocolFailure



- 1850                   ○ RequestReceiptFailure
- 1851                   ○ RequestAcceptanceFailure
- 1852                   ○ ResponseReceiptFailure
- 1853                   ○ ResponseAcceptanceFailure
- 1854                   ○ SignalTimeout
- 1855                   ○ ResponseTimeout
- 1856               • BusinessSuccess (isPositiveResponse=true or no isPositiveResponse
- 1857                   attribute)
- 1858               • BusinessFailure(isPositiveResponse=false)
- 1859               • Success (both protocol and business success)
- 1860               • Failure (AnyProtocolFailure or BusinessFailure).

1861

1862               BusinessFailure assumes that the transaction was successful from a “protocol”  
 1863               perspective, meaning that the state between the two parties could be effectively  
 1864               synchronized. However, the intent of the response was negative with respect to  
 1865               the request. As we mentioned earlier, this is an optional qualification of the  
 1866               response, agreed upon at design time, and some messages may not be  
 1867               qualifiable, i.e. they are neither positive or negative. The way business document  
 1868               specifications are designed allows to define two “logical” documents from the  
 1869               same physical document and a condition expression evaluated at run-time by the  
 1870               BSI. If the condition is true and isPositiveResponse = false, then the transaction  
 1871               ends in business failure based on the business document content. Of course  
 1872               entire documents can be tagged with isPositiveResponse=false, not just when  
 1873               they contain a particular field value.

1874

1875               It is required that each business transaction activity be designed such that there  
 1876               is at a minimum two transitions from the business transaction activity, one with a  
 1877               conditionGuard with a Success value, the other one with a Failure value. Even if  
 1878               in case of failure the transitions goes to the failure state of the collaboration.

## 1879   **7.7 Runtime Collaboration Semantics**

1880               The ebXML collaboration semantics contain a number of relationships between  
 1881               multipart collaborations and binary collaborations, between recursive layers of  
 1882               binary collaborations, and choreographies among transactions in binary  
 1883               collaborations. It is anticipated that over time BSI software will evolve to the point  
 1884               of monitoring and managing the state of a collaboration, similar to the way a BSI  
 1885               today is expected to manage the state of a transaction. For the immediate future,  
 1886               such capabilities are not expected and not required.

## 1887   **7.8 Where the ebXML Business Process Specification Schema** 1888   **May Be Implemented**

1889               The ebXML *Business Process Specification Schema* should be used wherever  
 1890               software is being specified to perform a role in an ebXML business collaboration.  
 1891               Specifically, the ebXML *Business Process Specification Schema* is intended to  
 1892               provide the business process and document specification for the formation of  
 1893               ebXML trading partner Collaboration Protocol Profiles and Agreements.

1894 However, the ebXML *Business Process Specification Schema* may be used to  
1895 specify any electronic commerce collaboration. It may also be used for non-  
1896 commerce collaborations, for instance in defining transactional collaborations  
1897 among non-profit organizations or internally in enterprises.

1898 Every BSI which is in the position of sending a signal or a document envelop  
1899 shall verify if sending this message will violate the business transaction  
1900 definitions and shall not send it if such a condition is detected. For instance  
1901 sending a signal or a response after a timeout has occurred is prohibited.  
1902 Similarly, sending a receipt on a document envelop which do not have the same  
1903 digest as the original document envelop is prohibited. Rather, the BSI should  
1904 send an exception back to the application initiated the particular message.

1905 Compliant BSIs do not need to support multiparty collaborations. It will be  
1906 required for further version of the specification.

1907 As of the current version, it is not requested that BSI be able to support multi-  
1908 party collaboration. The current specification does not support the notions of  
1909 context and correlation, which makes in hard to implement but for simple cases.  
1910 Future releases of this specification will provide the corresponding definitions.  
1911

## 1912 **7.9 Guidelines for Business Service Interface Interoperability**

1913 We have taken great care in this new version of the specification to distinguish  
1914 what is executable and computable versus general expressions written in text  
1915 and associated with model elements. In particular, we exclude, beginsWhen,  
1916 endsWhen, preCondition and postCondition from the responsibility of a BSI. In  
1917 the future these model constructs will become the responsibility of the BSI as we  
1918 identify an executable syntax.  
1919

1920 Another important point for interoperability is that the context of a binary  
1921 collaboration is limited to the document flows that are received or sent by the  
1922 BSI. In no case, the BSI could be in charge in looking up information in other  
1923 systems, internal or external to determine the result of condition expressions.  
1924

1925 A BSI is required to support two languages for the ConditionExpression element:  
1926 the XPath language, as well as the "DocumentEnvelopeLanguage". An XPath  
1927 expression may involve the content of any DocumentEnvelope received prior to  
1928 the transition within the scope of the current binary collaboration instance. The  
1929 "DocumentEnvelopeLanguage" is simply defined as the name or ID of a  
1930 document envelope.

1931  
1932

## 1933 **7.10 Collaboration and transaction well-formedness rules**

1934 The following rules should be used in addition to standard parsing to properly  
1935 constrain the values of the attributes of the elements in an ebXML Business  
1936 Process Specification.

- 1937 *Business Transaction*
- 1938 [0] If non-repudiation is required then the input or returned business  
1939 document must be a tamper-proofed entity.
- 1940 [1] If authorization is required then the input business document and  
1941 business signal must be an authenticated or a tamper proofed secure  
1942 entity.
- 1943 [2] The time to acknowledge receipt must be less than the time to  
1944 acknowledge acceptance if both properties have values.  
1945  
1946  $\text{timeToAcknowledgeReceipt} < \text{timeToAcknowledgeAcceptance}$
- 1947 [3] If the time to acknowledge acceptance is null then the time to perform  
1948 an activity must either be equal to or greater than the time to  
1949 acknowledge receipt.
- 1950 [4] The time to perform a transaction cannot be null if either the time to  
1951 acknowledge receipt or the time to acknowledge acceptance is not  
1952 null.
- 1953 [5] If non-repudiation of receipt is required then the time to acknowledge  
1954 receipt cannot be null.
- 1955 [6] The time to acknowledge receipt, time to acknowledge acceptance  
1956 and time to perform cannot all be zero.
- 1957 *RequestingBusinessActivity*
- 1958 [7] There must be one input transition whose source state vertex is an  
1959 initial pseudo state.
- 1960 [8] There must be one output transition whose target state vertex is a  
1961 final state specifying the state of the machine when the activity is  
1962 successfully performed.
- 1963 [9] There must be one output transition whose target state vertex is a  
1964 final state specifying the state of the machine when the activity is NOT  
1965 successfully performed due to a process control exception.
- 1966 [10] There must be one output transition whose target state vertex is a  
1967 final state specifying the state of the machine when the activity is NOT  
1968 successfully performed due to a business process exception.
- 1969 [11] There must be one output document flow from a requesting  
1970 business activity that in turn is the input to a responding business  
1971 activity.
- 1972 [12] There must be zero or one output document flow from a  
1973 responding business activity that in turn is the input to the requesting  
1974 business activity.

1975      *RespondingBusinessActivity*

1976 [13] There must be one input transition from a document flow that in  
1977 turn has one input transition from a requesting business activity.

1978 [14] There must be zero or one output transition to a document flow  
1979 that in turn has an output transition to a requesting business activity.

1980 *Business Collaboration*

1981 [15] A Business Partner Role cannot provide both the roles of the  
1982 same business transaction activity.

1983

1984

## 8 ebXML Business Process Specification Schema –

In this section we describe the XML Schema version of the Specification Schema.

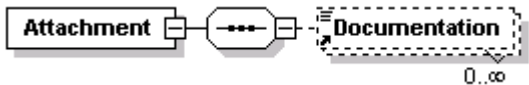
- An example XML Business Process Specification listed in Appendix A
- A table listing all the elements with definitions and parent/child relationships
- A table listing all the elements, each with a cross reference to the corresponding class in the UML version of the specification schema
- Rules about namespaces and element references

### 8.1 Documentation for the Schema

This section will document the Schema. The Schema has been derived from the UML model. The correlation between the UML classes and Schema elements will be shown separately later in this document.

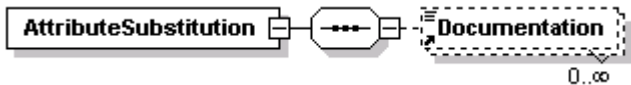
Elements:	Complex types	Attr. groups
<a href="#">Attachment</a>	<a href="#">BusinessAction</a>	<a href="#">documentSecurity</a>
<a href="#">AttributeSubstitution</a>	<a href="#">BusinessActivity</a>	<a href="#">name</a>
<a href="#">BinaryCollaboration</a>	<a href="#">RoleType</a>	
<a href="#">BusinessDocument</a>		
<a href="#">BusinessPartnerRole</a>		
<a href="#">BusinessTransaction</a>		
<a href="#">BusinessTransactionActivity</a>		
<a href="#">CollaborationActivity</a>		
<a href="#">ConditionExpression</a>		
<a href="#">Documentation</a>		
<a href="#">DocumentEnvelope</a>		
<a href="#">DocumentSubstitution</a>		
<a href="#">Failure</a>		
<a href="#">Fork</a>		
<a href="#">Include</a>		
<a href="#">Role</a>		
<a href="#">Join</a>		
<a href="#">MultiPartyCollaboration</a>		
<a href="#">Package</a>		
<a href="#">Performs</a>		
<a href="#">ProcessSpecification</a>		
<a href="#">RequestingBusinessActivity</a>		
<a href="#">RespondingBusinessActivity</a>		
<a href="#">Start</a>		
<a href="#">SubstitutionSet</a>		
<a href="#">Success</a>		
<a href="#">Transition</a>		

## 2001 8.1.1 Element Attachment

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>An optional attachment to a BusinessDocument in a DocumentEnvelope.</p> <p>Recommendation: Either use businessDocument +businessDocumentIDRef attributes OR use specification +mimeType attributes.</p>				
Children	<a href="#">Documentation</a>				
Used by	element <a href="#">DocumentEnvelope</a>				
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the attachment.
	nameID	xsd:ID			XML ID version of name
	businessDocument	xsd:string	required		A BusinessDocument can define an Attachment's type. If it is not of a defined Business Document, the mime type and specification attribute will be the only indication of its type.
	businessDocumentIDRef	xsd:IDREF		optional	The XML IDREF version of businessDocument
	specification	xsd:anyURI			A reference to an external source of description of this attachment.
	mimeType	xsd:string			Defines the valid MIME (Multipurpose Internet Mail Extensions) type of this Attachment. Example: 'application/pdf'
	isAuthenticated	xsd:NMTOKEN			There is a digital certificate associated with the document entity. This provides proof of the signer's identity.(See also section on Document Security)
	isConfidential	xsd:NMTOKEN		none	The information entity is encrypted so that unauthorized parties cannot view the information(See also section on Document Security)
	isTamperDetectable	xsd:NMTOKEN		none	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.(See also section on Document Security)
source	<pre> &lt;xsd:element name="Attachment"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;     &lt;xsd:attribute name="businessDocument" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="businessDocumentIDRef" type="xsd:IDREF"/&gt;     &lt;xsd:attribute name="specification" type="xsd:anyURI"/&gt;     &lt;xsd:attribute name="mimeType" type="xsd:string" use="optional"/&gt;     &lt;xsd:attributeGroup ref="documentSecurity"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2002

## 2003 8.1.2 Element: AttributeSubstitution

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	Attribute Substitution specifies that an attribute value should be used in place of some attribute value in an existing process specification.				
Children	<a href="#">Documentation</a>				
Used by	element <a href="#">SubstitutionSet</a>				
Attributes	Name	Type	Use	Default	Annotation
	attributeName	xsd:string	required		The name of an attribute of any element within the scope of the substitution set.
	value	xsd:string	required		The value, which shall replace the current value of the attribute.
Source	<pre> &lt;xsd:element name="AttributeSubstitution"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="attributeName" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="value" type="xsd:string" use="required"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2004

## 2005 8.1.3 Element: BinaryCollaboration

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>Binary Collaboration defines a protocol of interaction between two roles. One must be the initiating role, and one the responding role. Binary Collaboration is a choreographed set of states among collaboration roles. The activities of performing business transactions or other collaborations are a kind of state. Binary Collaboration choreographs one or more business transaction activities between two roles. Binary Collaboration is not an atomic transaction. A binary collaboration may be used within another binary collaboration via a collaboration activity.</p>				
Children	<a href="#">Documentation</a> <a href="#">Role</a> <a href="#">Start</a> <a href="#">BusinessTransactionActivity</a> <a href="#">CollaborationActivity</a> <a href="#">Transition</a> <a href="#">Success</a> <a href="#">Failure</a> <a href="#">Fork</a> <a href="#">Join</a>				
Used by	elements	<a href="#">Package</a> <a href="#">ProcessSpecification</a>			
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Binary Collaboration.
	nameID	xsd:ID			The XML ID version of name Recommendation: name ID should be globally unique identifier.
	pattern	xsd:anyURI			The optional reference to a pattern that this binary collaboration is based on.
	beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
	endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
	preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.



	postCondition      xsd:string  timeToPerform      xsd:duration  initiatingRoleID      xsd:IDREF      required	A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration. The period of time, starting upon initiation of the first activity, within which this entire collaboration must conclude. Reference to the role that initiates the transaction.
source	<pre> &lt;xsd:element name="BinaryCollaboration"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element name="Role" type="RoleType" minOccurs="2" maxOccurs="2"/&gt;       &lt;xsd:element ref="Start"/&gt;       &lt;xsd:choice maxOccurs="unbounded"&gt;         &lt;xsd:element ref="BusinessTransactionActivity"/&gt;         &lt;xsd:element ref="CollaborationActivity"/&gt;       &lt;/xsd:choice&gt;       &lt;xsd:element ref="Success" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Failure" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Transition" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Fork" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Join" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;     &lt;xsd:attribute name="pattern" type="xsd:anyURI"/&gt;     &lt;xsd:attribute name="beginsWhen" type="xsd:string"/&gt;     &lt;xsd:attribute name="endsWhen" type="xsd:string"/&gt;     &lt;xsd:attribute name="preCondition" type="xsd:string"/&gt;     &lt;xsd:attribute name="postCondition" type="xsd:string"/&gt;     &lt;xsd:attribute name="timeToPerform" type="xsd:duration"/&gt;     &lt;xsd:attribute name="initiatingRoleID" type="xsd:IDREF" use="required"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>	

2006  
2007  
2008

### 8.1.4 Element: BusinessDocument

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	BusinessDocument is a generic name of a document. A BusinessDocument may have one Condition Expression. This determines whether this is a valid business document for its envelope.				
Children	<a href="#">Documentation</a> <a href="#">ConditionExpression</a>				
Used by	elements	<a href="#">Package</a>	<a href="#">ProcessSpecification</a>		
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the generic name of the Business
	nameID	xsd:ID			XML ID version of name
	specificationLocation	xsd:anyURI			Reference to an external source of the schema

	specificationID      xsd:anyURI	Reference to an external source of the schema definition. This defines the absolute path including the element id within the schema definition that defines the type of this document. Use either specificationLocation or specificationID.
source	<pre> &lt;xsd:element name="BusinessDocument"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="ConditionExpression" minOccurs="0"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;     &lt;xsd:attribute name="specificationLocation" type="xsd:anyURI"/&gt;     &lt;xsd:attribute name="specificationID" type="xsd:anyURI"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>	

2009  
2010  
2011

### 8.1.5 Element: BusinessPartnerRole

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>A BusinessPartnerRole is the role played by a business partner in a MultiPartyCollaboration. A BusinessPartnerRole performs at most one Role in each of the Binary Collaborations that make up the Multiparty Collaboration.</p> <p>Wellformedness Rule: A partner must not perform both roles in a given business activity</p>				
Children	<a href="#">Documentation</a> <a href="#">Performs</a> <a href="#">Transition</a>				
Used by	element <a href="#">MultiPartyCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the role played by a partner in the overall multiparty business collaboration, e.g. customer or supplier.
	nameID	xsd:ID			The XML ID version of name
Source	<pre> &lt;xsd:element name="BusinessPartnerRole"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Performs" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Transition" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2012

## 2013 8.1.6 Element: BusinessTransaction

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>A business transaction is a set of business information and business signal exchanges amongst two commercial partners that must occur in an agreed format, sequence and time period. If any of the agreements are violated then the transaction is terminated and all business information and business signal exchanges must be discarded. Business Transactions can be formal as in the formation of on-line offer/acceptance commercial contracts and informal as in the distribution of product announcements. A BusinessTransaction can be performed by many BusinessTransactionActivities. A BusinessTransaction has exactly one RequestingBusinessActivity. A BusinessTransaction has exactly one RespondingBusinessActivity</p>				
Children	<a href="#">Documentation</a> <a href="#">RequestingBusinessActivity</a> <a href="#">RespondingBusinessActivity</a>				
Used by	elements	<a href="#">Package</a> <a href="#">ProcessSpecification</a>			
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Business Transaction.
	nameID	xsd:ID			The XML ID version of name. Recommendation: name ID should be globally unique identifier
	pattern	xsd:anyURI			The optional reference to a pattern that this transaction is based on
	isGuaranteedDeliveryRequired	xsd:boolean		false	Both partners must agree to use a transport that guarantees delivery
Source	<pre> &lt;xsd:element name="BusinessTransaction"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="RequestingBusinessActivity"/&gt;       &lt;xsd:element ref="RespondingBusinessActivity"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;     &lt;xsd:attribute name="pattern" type="xsd:anyURI"/&gt;     &lt;xsd:attribute name="isGuaranteedDeliveryRequired" type="xsd:boolean" default="false"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2014

## 2015 8.1.7 Element: BusinessTransactionActivity

2016

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
type	extension of <b>BusinessActivity</b>				
Description	<p>A business transaction activity defines the use of a business transaction within a binary collaboration. A business transaction activity is a business activity that executes a specified business transaction. More than one instance of the same business transaction activity can be open at one time if the isConcurrent property is true. A Role may not be both the requestor and the responder in a business transaction.</p>				

Children	<a href="#">Documentation</a>				
Used by	element <a href="#">BinaryCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the activity uniquely within the binary collaboration
	nameID	xsd:ID			The XML ID version of name
	businessTransaction	xsd:string	required		A reference, by name to the Business Transaction performed by this Business Transaction Activity
	businessTransactionIDRef	xsd:IDREF			The XML IDREF version of businessTransaction
	fromRole	xsd:string	required		The name of the initiating role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the requestor in the BusinessTransaction performed by this activity
	fromRoleIDRef	xsd:IDREF			The XML IDREF version of fromAuthorizedRole
	toRole	xsd:string	required		The name of the responding role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the responder in the BusinessTransaction performed by this activity
	toRoleIDRef	xsd:IDREF			The XML IDREF version of toAuthorizedRole
	isConcurrent	xsd:boolean		true	If the BusinessTransactionActivity is concurrent then more than one instance of the associated BusinessTransaction can be open the same time as part of the execution of this BusinessTransactionActivity
	isLegallyBinding	xsd:boolean		true	Defines whether the Business Transaction performed by this activity is intended by the trading parties to be binding. Default value is True.
	timeToPerform	xsd:duration			The period of time, starting upon the sending of the request, within which both partners agree to conclude the business transaction executed by this Business Transaction Activity.
	beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
	endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
	preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.
	postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration.
Source	<pre> &lt;xsd:element name="BusinessTransactionActivity"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:complexContent&gt;       &lt;xsd:extension base="BusinessActivity"&gt;         &lt;xsd:sequence&gt;           &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;/xsd:sequence&gt;         &lt;xsd:attribute name="businessTransaction" type="xsd:string" use="required"/&gt;         &lt;xsd:attribute name="businessTransactionIDRef" type="xsd:IDREF"/&gt;         &lt;xsd:attribute name="isConcurrent" type="xsd:boolean" default="true"/&gt;         &lt;xsd:attribute name="isLegallyBinding" type="xsd:boolean" default="true"/&gt;         &lt;xsd:attribute name="timeToPerform" type="xsd:duration"/&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:complexContent&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2017

2018

2019

## 8.1.8 Element: CollaborationActivity

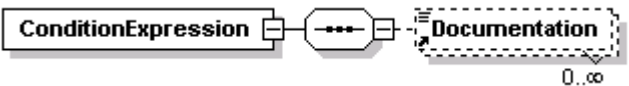
Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
type	extension of <b>BusinessActivity</b>				
Description	A collaboration activity is the activity of performing a binary collaboration within another binary collaboration				
Children	<a href="#">Documentation</a>				
Used by	element	<a href="#">BinaryCollaboration</a>			
Attributes	Name	Type	Use	Default	Annotation
	nameID	xsd:ID			ID of the name
	name	xsd:string	required		Defines the name of the activity uniquely within the binary collaboration
	fromRole	xsd:string	required		The name of the initiating role in the Collaboration Activity. This must match one of the AuthorizedRoles in the parent binary collaboration and will become the initiator in the BinaryCollaboration performed by this activity
	fromRoleIDRef	xsd:IDREF			The XML IDREF version of fromAuthorizedRole
	toRole	xsd:string	required		The name of the responding role in the Collaboration Activity. This must match one of the AuthorizedRoles in the parent binary collaboration and will become the responder in the BinaryCollaboration performed by this activity
	toRoleIDRef	xsd:IDREF			The XML IDREF version of toAuthorizedRole
	binaryCollaboration	xsd:string	required		A reference, by name, to the Binary Collaboration performed by this Collaboration Activity
	binaryCollaborationIDRef	xsd:IDREF			The XML IDREF version of binaryCollaboration
	beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
	endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
	preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.
	postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration.
Source	<pre> &lt;xsd:element name="CollaborationActivity"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:complexContent&gt;       &lt;xsd:extension base="BusinessActivity"&gt;         &lt;xsd:sequence&gt;           &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;/xsd:sequence&gt;         &lt;xsd:attribute name="binaryCollaboration" type="xsd:string" use="required"/&gt;         &lt;xsd:attribute name="binaryCollaborationIDRef" type="xsd:IDREF"/&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:complexContent&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2020

2021


2022  
2023

## 8.1.9 Element: ConditionExpression

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	Condition Expression is an expression that can be evaluated to TRUE or FALSE.				
Children	<a href="#">Documentation</a>				
Used by	elements <a href="#">BusinessDocument</a> <a href="#">Failure</a> <a href="#">Success</a> <a href="#">Transition</a>				
Attributes	Name	Type	Use	Default	Annotation
	expressionLanguage expression	xsd:string xsd:string	required required		The language of the expression, e.g. XPATH An expression whose evaluation results in TRUE or FALSE. For a transition, this determines whether this transition should happen or not. For a business document, this determines whether this is a valid business document for its envelope. The expression can refer to the name or content of the most recent DocumentEnvelope or content of documents within it.
Source	<pre>&lt;xsd:element name="ConditionExpression"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="expressionLanguage" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="expression" type="xsd:string" use="required"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt;</pre>				

2024  
2025  
2026  
2027

## 8.1.10 Element: Documentation

Diagram		
Namespace	http://www.ebxml.org/BusinessProcess	
type	extension of <b>xsd:string</b>	
Description	Defines user documentation for any element. Must be the first element of its container. Documentation can be either inline PCDATA and/or a URI to where more complete documentation is to be found	
Used by	elements	<a href="#">Attachment</a> <a href="#">AttributeSubstitution</a> <a href="#">BinaryCollaboration</a> <a href="#">BusinessDocument</a> <a href="#">BusinessPartnerRole</a> <a href="#">BusinessTransaction</a> <a href="#">BusinessTransactionActivity</a> <a href="#">CollaborationActivity</a> <a href="#">ConditionExpression</a> <a href="#">DocumentEnvelope</a> <a href="#">DocumentSubstitution</a> <a href="#">Failure</a> <a href="#">Fork</a> <a href="#">Include</a> <a href="#">Join</a> <a href="#">MultiPartyCollaboration</a> <a href="#">Package</a> <a href="#">Performs</a> <a href="#">ProcessSpecification</a> <a href="#">Start</a> <a href="#">SubstitutionSet</a> <a href="#">Success</a> <a href="#">Transition</a>
	complexTypees	<a href="#">BusinessAction</a> <a href="#">RoleType</a>
Source	<pre>&lt;xsd:element name="Documentation"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:simpleContent&gt;       &lt;xsd:extension base="xsd:string"&gt;         &lt;xsd:attribute name="uri" type="xsd:anyURI"/&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:simpleContent&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt;</pre>	

2028  
2029  
2030

## 8.1.11 Element: DocumentEnvelope

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>A DocumentEnvelope is what conveys business information between the two roles in a business transaction. One DocumentEnvelope conveys the request from the requesting role to the responding role, and another DocumentEnvelope conveys the response (if any) from the responding role back to the requesting role. A documentEnvelope contains exactly one primary Business document. It contains an optional set of attachments related to primary document.</p> <p>Wellformedness Rules: A Document Envelope is associated with exactly one requesting and one responding activity.</p> <p>IsPositiveResponse is not a relevant parameter on a DocumentEnvelope sent by a requesting activity</p>				
Children	<a href="#">Documentation</a> <a href="#">Attachment</a>				
Used by	elements <a href="#">RequestingBusinessActivity</a> <a href="#">RespondingBusinessActivity</a>				
attributes	Name	Type	Use	Default	Annotation
	name	xsd:string			Defines Name of the DocumentEnvelope
	nameID	xsd:ID			Defines ID of the DocumentEnvelope
	businessDocument	xsd:string	required		The name of the business document.
	businessDocumentIDRef	xsd:IDREF			The XML IDREF version of businessDocument
	isPositiveResponse	xsd:boolean			TRUE or FALSE. If TRUE this DocumentEnvelope is intended as a positive response to the request. The value for this parameter supplied for a DocumentEnvelope is an assertion by the sender of the DocumentEnvelope regarding its intent for the transaction to which it relates, but does not bind the recipient, or override the computation of transactional success or failure using the transaction's guard expressions. In some situations this could be an XPath expression that interrogates the BusinessDocument in the envelope. IsPositiveResponse is only relevant for responses, and is ignored in requests.
	isAuthenticated	xsd:NMTOKEN		none	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".
	isConfidential	xsd:NMTOKEN		none	The information entity is encrypted so that unauthorized parties cannot view the information.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".
	isTamperDetectable	xsd:NMTOKEN		none	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.(See also section on Document Security)The value of the attribute, if other than "none" should be interpreted as "at least value".
Source	<xsd:element name="DocumentEnvelope">				

```

<xsd:complexType>
  <xsd:sequence>
    <xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element ref="Attachment" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attributeGroup ref="name"/>
  <xsd:attribute name="businessDocument" type="xsd:string" use="required"/>
  <xsd:attribute name="businessDocumentIDRef" type="xsd:IDREF"/>
  <xsd:attribute name="isPositiveResponse" type="xsd:boolean"/>
  <xsd:attributeGroup ref="documentSecurity"/>
</xsd:complexType>
</xsd:element>

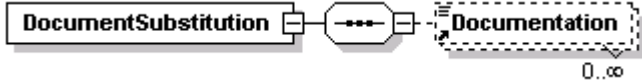
```

2031

2032

2033

### 8.1.12 Element: DocumentSubstitution

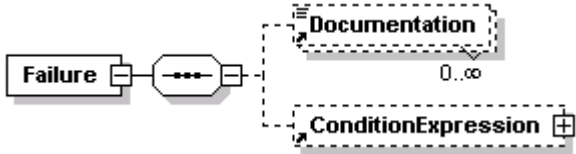
diagram					
namespace	http://www.ebxml.org/BusinessProcess				
Description	DocumentSubstitution specifies a document that should be used in place of a document in an existing process specification.				
children	<a href="#">Documentation</a>				
used by	element	<a href="#">SubstitutionSet</a>			
attributes	Name	Type	Use	Default	Annotation
	originalBusinessDocument	xsd:string	required		The name of a business document within the scope of the substitution set.
	originalBusinessDocumentID	xsd:ID			The ID of the business document.
	substituteBusinessDocumentLocation	xsd:anyURI	required		The location of the document which shall replace the current document.
	substituteBusinessDocumentID	xsd:anyURI			The ID of the replacement document.
source	<pre> &lt;xsd:element name="DocumentSubstitution"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="originalBusinessDocument" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="originalBusinessDocumentID" type="xsd:ID"/&gt;     &lt;xsd:attribute name="substituteBusinessDocumentLocation" type="xsd:anyURI" use="required"/&gt;     &lt;xsd:attribute name="substituteBusinessDocumentID" type="xsd:anyURI"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2034

2035

2036

### 8.1.13 Element: Failure


Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>Defines the unsuccessful conclusion of a binary collaboration as a transition from an activity.</p> <p>Wellformedness Rules: Every Binary Collaboration should have at least one failure.</p>				



Children	<a href="#">Documentation</a> <a href="#">ConditionExpression</a>				
Used by	element <a href="#">BinaryCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	nameID	xsd:ID			Defines ID of the Failure
	fromBusinessState	xsd:string	required		The name of the activity from which this indicates a transition to unsuccessful conclusion of the BusinessTransaction or BinaryCollaboration
	fromBusinessStateIDRef	xsd:IDREF			The XML IDREF version of fromBusinessState
	conditionGuard	xsd:NMTOKEN			The condition that guards this transition
Source	<pre> &lt;xsd:element name="Failure"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="ConditionExpression" minOccurs="0"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="nameID" type="xsd:ID"/&gt;     &lt;xsd:attribute name="fromBusinessState" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="fromBusinessStateIDRef" type="xsd:IDREF"/&gt;     &lt;xsd:attribute name="conditionGuard"&gt;       &lt;xsd:simpleType&gt;         &lt;xsd:restriction base="xsd:NMTOKEN"&gt;           &lt;xsd:enumeration value="ProtocolSuccess"/&gt;           &lt;xsd:enumeration value="AnyProtocolFailure"/&gt;           &lt;xsd:enumeration value="RequestReceiptFailure"/&gt;           &lt;xsd:enumeration value="RequestAcceptanceFailure"/&gt;           &lt;xsd:enumeration value="ResponseReceiptFailure"/&gt;           &lt;xsd:enumeration value="ResponseAcceptanceFailure"/&gt;           &lt;xsd:enumeration value="SignalTimeout"/&gt;           &lt;xsd:enumeration value="ResponseTimeout"/&gt;           &lt;xsd:enumeration value="BusinessSuccess"/&gt;           &lt;xsd:enumeration value="BusinessFailure"/&gt;           &lt;xsd:enumeration value="Success"/&gt;           &lt;xsd:enumeration value="Failure"/&gt;         &lt;/xsd:restriction&gt;       &lt;/xsd:simpleType&gt;     &lt;/xsd:attribute&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

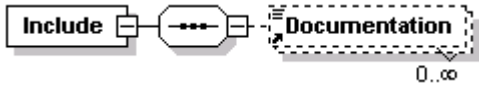
2037  
2038  
2039  
2040

## 8.1.14 Element: Fork


Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	A Fork is a state with one inbound transition and multiple outbound transitions. All activities pointed to by the outbound transitions are assumed to happen in parallel or exclusive or.				
Children	<a href="#">Documentation</a>				
Used by	element <a href="#">BinaryCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Fork state
	nameID	xsd:ID			The XML ID version of name
	type	xsd:NMTOKEN	required	All	All : all activities will run in parallel. XOR:

	timeToPerform    xsd:duration    optional	Only one of the possible activities will run. timeToPerform attribute on the Fork element may be used to specify that the business activities between the Fork and the Join shall be executed within the specified duration otherwise, the state of the collaboration will automatically advance to the join.
Source	<pre> &lt;xsd:element name="Fork"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;     &lt;xsd:attribute name="type" use="optional" default="All"&gt;       &lt;xsd:simpleType&gt;         &lt;xsd:restriction base="xsd:NMTOKEN"&gt;           &lt;xsd:enumeration value="All"/&gt;           &lt;xsd:enumeration value="XOR"/&gt;         &lt;/xsd:restriction&gt;       &lt;/xsd:simpleType&gt;     &lt;/xsd:attribute&gt;     &lt;xsd:attribute name="timeToPerform" type="xsd:duration" use="optional"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>	

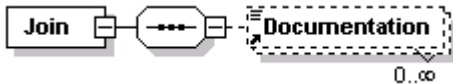
### 8.1.15 Element: Include

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	Includes another process specification document and merges that specification with the current specification. Any elements of the same name and in the same name scope must have exactly the same specification except that packages may have additional content. Documents are merged based on name scope. A name in an included package will be indistinguishable from a name in the base document.				
Children	<a href="#">Documentation</a>				
Used by	elements	<a href="#">Package ProcessSpecification</a>			
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Name of the included specification
	uuid	xsd:ID	required		Unique identifier of the included specification
	uri	xsd:anyURI	required		URI of the included specification
	version	xsd:string	required		Version of the included specification
Source	<pre> &lt;xsd:element name="Include"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="name" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="uuid" type="xsd:ID" use="required"/&gt;     &lt;xsd:attribute name="uri" type="xsd:anyURI" use="required"/&gt;     &lt;xsd:attribute name="version" type="xsd:string" use="required"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2047 8.1.16 Element:Role  
2048

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
type	<a href="#">RoleType</a>				
Children	<a href="#">Documentation</a>				
Used by	element <a href="#">BinaryCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Initiating Role
	nameID	xsd:ID			XML ID version the name.
Source	<xsd:element name="Role" type="RoleType" minOccurs="2" maxOccurs="2"/>				

2049 8.1.17 Element: Join  
2050

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	A business state where an activity is waiting for the completion of one or more other activities. Defines the point where previously forked activities join up again.				
Children	<a href="#">Documentation</a>				
Used by	element <a href="#">BinaryCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the Join state.
	nameID	xsd:ID			The XML ID version of name
	waitForAll	xsd:boolean		true	Boolean value indicating if this Join state should wait for all incoming transitions to complete. If TRUE, wait for all, if False proceed on first incoming transition.
Source	<pre>&lt;xsd:element name="Join"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;     &lt;xsd:attribute name="waitForAll" type="xsd:boolean" default="true"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt;</pre>				

2051  
2052

## 2053 8.1.18 Element: MultiPartyCollaboration

diagram					
namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>A Multiparty Collaboration is a synthesis of Binary Collaborations. A Multiparty Collaboration consists of a number of Business Partner Roles each playing roles in binary collaborations with each other.</p> <p>Wellformedness Rules: All multiparty collaborations must be synthesized from binary collaborations</p>				
children	<a href="#">Documentation</a> <a href="#">BusinessPartnerRole</a>				
used by	elements	<a href="#">Package</a> <a href="#">ProcessSpecification</a>			
attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the MultiPartyCollaboration
	nameID	xsd:ID			The XML ID version of name
source	<pre> &lt;xsd:element name="MultiPartyCollaboration"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="BusinessPartnerRole" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

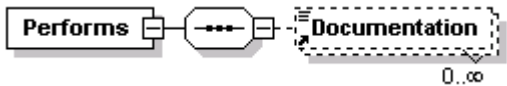
2054  
2055

## 2056 8.1.19 Element: Package

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Children	<a href="#">Documentation</a> <a href="#">Include</a> <a href="#">Package</a> <a href="#">BusinessDocument</a> <a href="#">BusinessTransaction</a> <a href="#">BinaryCollaboration</a> <a href="#">MultiPartyCollaboration</a>				
Description	Defines a hierarchical name scope containing reusable elements.				
Used by	elements	<a href="#">Package</a> <a href="#">ProcessSpecification</a>			
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Name of the package
	nameID	xsd:ID			XML ID version of name
Source	<pre> &lt;xsd:element name="Package"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Include" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:choice minOccurs="0" maxOccurs="unbounded"&gt;         &lt;xsd:element ref="Package" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="BusinessDocument" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="BusinessTransaction" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="BinaryCollaboration" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="MultiPartyCollaboration" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;/xsd:choice&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2057  
2058

2059 8.1.20 Element: Performs

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>Performs is an explicit modeling of the relationship between a BusinessPartnerRole and the Roles it plays. This specifies the use of an Authorized Role within a multiparty collaboration.</p> <p>Wellformedness Rules: For every Performs performing an AuthorizedRole there must be a Performs that performs the opposing AuthorizedRole, otherwise the MultiParty Collaboration is not complete.</p>				
Children	<a href="#">Documentation</a>				
Used by	element <a href="#">BusinessPartnerRole</a>				
Attributes	Name	Type	Use	Default	Annotation
	nameID	xsd:ID			Defines ID of the DocumentEnvelope
	role	xsd:string	optional		The Role that will be performed by the Business PartnerRole, qualified with the name of the BinaryCollaboration
	roleIDRef	xsd:IDREF	optional		The XML IDREF version of Role
Source	<pre>&lt;xsd:element name="Performs"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="nameID" type="xsd:ID"/&gt;     &lt;xsd:attribute name="role" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="roleIDRef" type="xsd:IDREF" use="required"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt;</pre>				

2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068

## 2069 8.1.21 Element: ProcessSpecification

Diagram						
Namespace	http://www.ebxml.org/BusinessProcess					
Description	Root element of a process specification document that has a globally unique identity.					
children	<a href="#">Documentation</a> <a href="#">Include</a> <a href="#">Package</a> <a href="#">SubstitutionSet</a> <a href="#">BusinessDocument</a> <a href="#">BusinessTransaction</a> <a href="#">BinaryCollaboration</a> <a href="#">MultiPartyCollaboration</a>					
attributes	Name	Type	Use	Default	Annotation	
	name	xsd:string	required		Defines the name of the ProcessSpecification element.	
	uuid	xsd:anyURI	required		The ID of the ProcessSpecification element.	
	version	xsd:string	required		Version of the specification.	
identity constraints	unique	Name	Refer	Selector	Field(s)	
		ProcessSpecification-ID		.	uuid	
source	<pre>&lt;xsd:element name="ProcessSpecification"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="Include" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:choice minOccurs="0" maxOccurs="unbounded"&gt;         &lt;xsd:element ref="Package" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="SubstitutionSet" minOccurs="0"/&gt;         &lt;xsd:element ref="BusinessDocument" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="BusinessTransaction" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="BinaryCollaboration" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;xsd:element ref="MultiPartyCollaboration" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;/xsd:choice&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="name" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="uuid" type="xsd:anyURI" use="required"/&gt;     &lt;xsd:attribute name="version" type="xsd:string" use="required"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt;</pre>					

```
<xsd:unique name="ProcessSpecification-ID">
<xsd:selector xpath="."/>
<xsd:field xpath="uuid"/>
</xsd:unique>
</xsd:element>
```

2070  
2071  
2072

8.1.22 Element: RequestingBusinessActivity

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	A RequestingBusinessActivity is a Business Action that is performed by the requesting role within a Business Transaction. It specifies the Document Envelope which will carry the request.				
type	extension of <a href="#">BusinessAction</a>				
Children	<a href="#">Documentation</a> <a href="#">DocumentEnvelope</a>				
Used by	element <a href="#">BusinessTransaction</a>				
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the RequestingBusinessTransaction
	nameID	xsd:ID			The XML ID version of name
	isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originator against a list of authorized originators. This parameter is specified on the sending side. (See also section on action security)
	isIntelligibleCheckRequired	xsd:boolean		false	Receiving party must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt. This parameter is specified on the sending side. (See also section on core transaction semantics)



	retryCount                      xsd:int	Business Level Retries - Aligns with UMM. The message id be different when a retried from business process level and the receiving application will be responsible for detecting for business level duplicates using the unique id((like a purchase order number) ) in the business information.
Source	<pre> &lt;xsd:element name="RequestingBusinessActivity"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:complexContent&gt;       &lt;xsd:extension base="BusinessAction"&gt;         &lt;xsd:sequence&gt;           &lt;xsd:element ref="DocumentEnvelope"/&gt;         &lt;/xsd:sequence&gt;         &lt;xsd:attribute name="retryCount" type="xsd:int"/&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:complexContent&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>	

2073  
 2074  
 2075  
 2076  
 2077  
 2078  
 2079  
 2080  
 2081  
 2082  
 2083  
 2084  
 2085  
 2086  
 2087  
 2088  
 2089  
 2090  
 2091  
 2092  
 2093


### 8.1.23 Element: RespondingBusinessActivity

Diagram		
Namespace	http://www.ebxml.org/BusinessProcess	
Description	A RespondingBusinessActivity is a Business Action that is performed by the responding role within a Business Transaction. It specifies the Document Envelope which will carry the response. There may be multiple possible response Document Envelopes defined, but only one of them will be sent during an actual transaction instance.	
type	extension of <a href="#">BusinessAction</a>	

Children	<a href="#">Documentation</a> <a href="#">DocumentEnvelope</a>				
Used by	element	<a href="#">BusinessTransaction</a>			
Attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the RespondingBusinessTransaction
	nameID	xsd:ID			The XML ID version of name
	isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originator against a list of authorized originators. This parameter is specified on the sending side. (See also section on action security)
	isIntelligibleCheckRequired	xsd:boolean		false	Receiving party must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt. This parameter is specified on the sending side. (See also section on core transaction semantics)
	isNonRepudiationReceiptRequired	xsd:boolean		false	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt. This parameter is specified on the sending side. (See also section on nonrepudiation)
	isNonRepudiationRequired	xsd:boolean		false	Requires the sending parties to save copies of the transacted documents before sending them. (See also section on nonrepudiation)
	timeToAcknowledgeReceipt	xsd:duration			The time the receiving party has to acknowledge receipt of a business document. This parameter is specified on the sending side. (See also section on core transaction semantics)
Source					
	<pre> &lt;xsd:element name="RespondingBusinessActivity"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:complexContent&gt;       &lt;xsd:extension base="BusinessAction"&gt;         &lt;xsd:sequence&gt;           &lt;xsd:element ref="DocumentEnvelope" minOccurs="0" maxOccurs="unbounded"/&gt;         &lt;/xsd:sequence&gt;       &lt;/xsd:extension&gt;     &lt;/xsd:complexContent&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

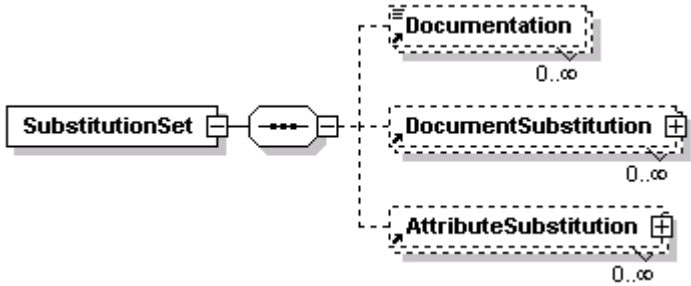
2094  
2095  
2096  
2097  
2098  
2099  
2100

## 2101 8.1.24 Element: Start

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	The starting state for an Binary Collaboration. A Binary Collaboration should have only one starting activity.				
Children	<a href="#">Documentation</a>				
Used by	element <a href="#">BinaryCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	toBusinessState	xsd:string	required		The name of an activity which an allowable starting point for this for BinaryCollaboration
	toBusinessStateIDRef	xsd:IDREF			The XML IDREF version of toBusinessState
Source	<pre> &lt;xsd:element name="Start"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="toBusinessState" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="toBusinessStateIDRef" type="xsd:IDREF"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>				

2102  
2103  
2104  
2105  
2106  
2107

## 2108 8.1.25 Element: SubstitutionSet

diagram					
namespace	http://www.ebxml.org/BusinessProcess				
Description	A Substitution Set is a container for one or more AttributeSubstitution and/or DocumentSubstitution elements. The entire SubstitutionSet specifies document or attribute values that should be used in place of some documents and attribute values in an existing process specification.				
children	<a href="#">Documentation</a> <a href="#">DocumentSubstitution</a> <a href="#">AttributeSubstitution</a>				
used by	element <a href="#">ProcessSpecification</a>				
attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Name of the substitution set.
	nameId	xsd:ID	required		The ID of the substitution set.
	applyToScope	xsd:string	required		Specifies the path to attributes or documents that are to be substituted for.

source	<pre> &lt;xsd:element name="SubstitutionSet"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="DocumentSubstitution" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="AttributeSubstitution" minOccurs="0" maxOccurs="unbounded"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attributeGroup ref="name"/&gt;     &lt;xsd:attribute name="applyToScope" type="xsd:string" use="required"/&gt;   &lt;/xsd:complexType&gt; &lt;/xsd:element&gt; </pre>
--------	---

2109

2110

2111

## 8.1.26 Element: Success

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	<p>Defines the successful conclusion of a binary collaboration as a transition from an activity.</p> <p>Wellformedness Rules: Every activity Binary Collaboration should have at least one success.</p>				
Children	<a href="#">Documentation</a> <a href="#">ConditionExpression</a>				
Used by	element <a href="#">BinaryCollaboration</a>				
Attributes	Name	Type	Use	Default	Annotation
	nameID	xsd:ID			Defines ID of the DocumentEnvelope
	fromBusinessState	xsd:string	required		The name of the activity from which this indicates a transition to successful conclusion of the BusinessTransaction or BinaryCollaboration
	fromBusinessStateIDRef	xsd:IDREF			ID of the business state
	conditionGuard	xsd:NMTOKEN			The condition that guards this transition
source	<pre> &lt;xsd:element name="Success"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="ConditionExpression" minOccurs="0"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="nameID" type="xsd:ID"/&gt;     &lt;xsd:attribute name="fromBusinessState" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="fromBusinessStateIDRef" type="xsd:IDREF"/&gt;     &lt;xsd:attribute name="conditionGuard"/&gt;   &lt;/xsd:complexType&gt;   &lt;xsd:simpleType&gt;     &lt;xsd:restriction base="xsd:NMTOKEN"&gt;       &lt;xsd:enumeration value="ProtocolSuccess"/&gt;       &lt;xsd:enumeration value="AnyProtocolFailure"/&gt;       &lt;xsd:enumeration value="RequestReceiptFailure"/&gt;       &lt;xsd:enumeration value="RequestAcceptanceFailure"/&gt;       &lt;xsd:enumeration value="ResponseReceiptFailure"/&gt;       &lt;xsd:enumeration value="ResponseAcceptanceFailure"/&gt;       &lt;xsd:enumeration value="SignalTimeout"/&gt;       &lt;xsd:enumeration value="ResponseTimeout"/&gt;       &lt;xsd:enumeration value="BusinessSuccess"/&gt;       &lt;xsd:enumeration value="BusinessFailure"/&gt;       &lt;xsd:enumeration value="Success"/&gt;       &lt;xsd:enumeration value="Failure"/&gt;     &lt;/xsd:restriction&gt;   &lt;/xsd:simpleType&gt; &lt;/xsd:element&gt; </pre>				

```

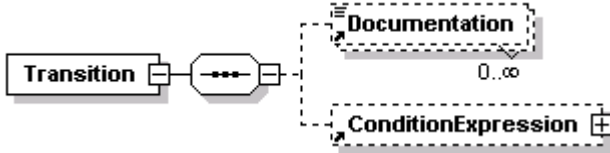
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>

```

2112

2113

## 8.1.27 Element: Transition

Diagram					
Namespace	http://www.ebxml.org/BusinessProcess				
Description	A transition is a transition between two business states in a binary collaboration. □Choreography is expressed as transitions between business states. Transition to the same state is allowed.				
Children	<a href="#">Documentation</a> <a href="#">ConditionExpression</a>				
Used by	elements	<a href="#">BinaryCollaboration</a> <a href="#">BusinessPartnerRole</a>			
Attributes	Name	Type	Use	Default	Annotation
	nameID	xsd:ID			Defines ID of the DocumentEnvelope
	onInitiation	xsd:boolean		false	This specifies this is a nested BusinessTransactionActivity and that upon receipt of the request in the associated transaction a second activity is performed before returning to the transaction to send the response back to the original requestor
	fromBusinessState	xsd:string	required		The name of the state transitioned from.
	fromBusinessStateIDRef	xsd:IDREF	optional		The XML IDREF version of fromBusinessState
	toBusinessState	xsd:string	required		The name of the state transitioned to
	toBusinessStateIDRef	xsd:IDREF	optional		The XML IDREF version of toBusinessState
	conditionGuard	xsd:NMTOKEN			A reference to the status of the previous transaction. A fixed value of Success, BusinessFailure, TechnicalFailure, or AnyFailure
source	<pre> &lt;xsd:element name="Transition"&gt;   &lt;xsd:complexType&gt;     &lt;xsd:sequence&gt;       &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;       &lt;xsd:element ref="ConditionExpression" minOccurs="0"/&gt;     &lt;/xsd:sequence&gt;     &lt;xsd:attribute name="nameID" type="xsd:ID"/&gt;     &lt;xsd:attribute name="onInitiation" type="xsd:boolean" default="false"/&gt;     &lt;xsd:attribute name="fromBusinessState" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="fromBusinessStateIDRef" type="xsd:IDREF" use="optional"/&gt;     &lt;xsd:attribute name="toBusinessState" type="xsd:string" use="required"/&gt;     &lt;xsd:attribute name="toBusinessStateIDRef" type="xsd:IDREF" use="optional"/&gt;     &lt;xsd:attribute name="conditionGuard"/&gt;   &lt;/xsd:complexType&gt;   &lt;xsd:restriction base="xsd:NMTOKEN"&gt;     &lt;xsd:enumeration value="ProtocolSuccess"/&gt;     &lt;xsd:enumeration value="AnyProtocolFailure"/&gt;     &lt;xsd:enumeration value="RequestReceiptFailure"/&gt;     &lt;xsd:enumeration value="RequestAcceptanceFailure"/&gt;     &lt;xsd:enumeration value="ResponseReceiptFailure"/&gt;     &lt;xsd:enumeration value="ResponseAcceptanceFailure"/&gt;     &lt;xsd:enumeration value="SignalTimeout"/&gt;     &lt;xsd:enumeration value="ResponseTimeout"/&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:element&gt; </pre>				

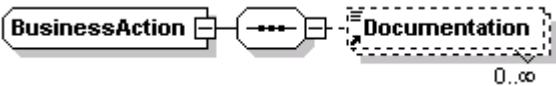
```

<xsd:enumeration value="BusinessSuccess"/>
<xsd:enumeration value="BusinessFailure"/>
<xsd:enumeration value="Success"/>
<xsd:enumeration value="Failure"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:complexType>
</xsd:element>

```

2114  
2115  
2116

## 8.1.28 complexType BusinessAction

diagram					
namespace	http://www.ebxml.org/BusinessProcess				
description					
children	<a href="#">Documentation</a>				
used by	elements <a href="#">RequestingBusinessActivity</a> <a href="#">RespondingBusinessActivity</a>				
attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the RequestingBusinessTransaction or RespondingBusinessTransaction depending on the subtype
	nameID	xsd:ID			Defines the nameID
	isAuthorizationRequired	xsd:boolean		false	Receiving party must validate identity of originator against a list of authorized originators. This parameter is specified on the sending side.
	isIntelligibleCheckRequired	xsd:boolean		false	Receiving party must check that a requesting document is not garbled (unreadable, unintelligible) before sending acknowledgement of receipt. This parameter is specified on the sending side.
	isNonRepudiationReceiptRequired	xsd:boolean		false	Requires the receiving party to return a signed receipt, and the original sender to save copy of the receipt. This parameter is specified on the sending side. (See also section on nonrepudiation)
	isNonRepudiationRequired	xsd:boolean		false	Requires the sending parties to save copies of the transacted documents before sending them(See also section on nonrepudiation)
	timeToAcknowledgeReceipt	xsd:duration			The time a receiving role has to acknowledge receipt of a business document. This parameter is specified on the sending side.
	timeToAcknowledgeAcceptance	xsd:duration			The time a responding role has to non-substantively acknowledge business acceptance of a business document. This parameter is specified on the requesting side.(See also section on core transaction semantics)
source	<pre> &lt;xsd:complexType name="BusinessAction"&gt;   &lt;xsd:sequence&gt;     &lt;xsd:element ref="Documentation" minOccurs="0" maxOccurs="unbounded"/&gt;   &lt;/xsd:sequence&gt;   &lt;xsd:attributeGroup ref="name"/&gt; </pre>				

```

<xsd:attribute name="isAuthorizationRequired" type="xsd:boolean" default="false"/>
<xsd:attribute name="isIntelligibleCheckRequired" type="xsd:boolean" default="false"/>
<xsd:attribute name="isNonRepudiationRequired" type="xsd:boolean" default="false"/>
<xsd:attribute name="isNonRepudiationReceiptRequired" type="xsd:boolean" default="false"/>
<xsd:attribute name="timeToAcknowledgeReceipt" type="xsd:duration"/>
<xsd:attribute name="timeToAcknowledgeAcceptance" type="xsd:duration"/>
</xsd:complexType>

```


2117  
2118  
2119

## 8.1.29 complexType BusinessActivity

diagram					
namespace	http://www.ebxml.org/BusinessProcess				
used by	elements <a href="#">BusinessTransactionActivity</a> <a href="#">CollaborationActivity</a>				
attributes	Name	Type	Use	Default	Annotation
	name	xsd:string	required		Defines the name of the activity uniquely within the binary collaboration
	nameID	xsd:ID			The XML ID version of name
	fromRole	xsd:string	required		The name of the initiating role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the requestor in the BusinessTransaction performed by this activity
	fromRoleIDRef	xsd:IDREF			The XML IDREF version of fromAuthorizedRole
	toRole	xsd:string	required		The name of the responding role in Business Transaction Activity. This must match one of the AuthorizedRoles in the binary collaboration and will become the responder in the BusinessTransaction performed by this activity
	toRoleIDRef	xsd:IDREF			The XML IDREF version of toAuthorizedRole
	beginsWhen	xsd:string			A description of an event external to the collaboration that normally causes this collaboration to commence.
	endsWhen	xsd:string			A description of an event external to this collaboration that normally causes this collaboration to conclude.
	preCondition	xsd:string			A description of a state external to this collaboration that is required before this collaboration can commence.
	postCondition	xsd:string			A description of a state that does not exist before the execution of this collaboration but will exist as a result of the execution of this collaboration.
source	<pre> &lt;xsd:complexType name="BusinessActivity"&gt;   &lt;xsd:attributeGroup ref="name"/&gt;   &lt;xsd:attribute name="fromRole" type="xsd:string" use="required"/&gt;   &lt;xsd:attribute name="fromRoleIDRef" type="xsd:IDREF"/&gt;   &lt;xsd:attribute name="toRole" type="xsd:string" use="required"/&gt;   &lt;xsd:attribute name="toRoleIDRef" type="xsd:IDREF"/&gt;   &lt;xsd:attribute name="beginsWhen" type="xsd:string"/&gt;   &lt;xsd:attribute name="endsWhen" type="xsd:string"/&gt;   &lt;xsd:attribute name="preCondition" type="xsd:string"/&gt;   &lt;xsd:attribute name="postCondition" type="xsd:string"/&gt; &lt;/xsd:complexType&gt; </pre>				

2120

## 2121 8.1.30 complexType RoleType

diagram						
namespace	http://www.ebxml.org/BusinessProcess					
children	<a href="#">Documentation</a>					
used by	elements <b>Role</b>					
attributes	Name name nameID	Type xsd:string xsd:ID	Use required	Default	Fixed	Annotation
source	<pre>&lt;xsd:complexType name="RoleType"&gt;   &lt;xsd:sequence&gt;     &lt;xsd:element ref="Documentation" minOccurs="0"/&gt;   &lt;/xsd:sequence&gt;   &lt;xsd:attribute name="name" type="xsd:string" use="required"/&gt;   &lt;xsd:attribute name="nameID" type="xsd:ID" use="required"/&gt; &lt;/xsd:complexType&gt;</pre>					

2122

2123

## 2124 8.1.31 attributeGroup documentSecurity

namespace	http://www.ebxml.org/BusinessProcess					
used by	elements <a href="#">Attachment</a> <a href="#">DocumentEnvelope</a>					
attributes	Name	Type	Use	Default	Annotation	
	isAuthenticated	xsd:NMTOKEN		none	There is a digital certificate associated with the document entity. This provides proof of the signer's identity.	
	isConfidential	xsd:NMTOKEN		none	The information entity is encrypted so that unauthorized parties cannot view the information.	
	isTamperDetectable	xsd:NMTOKEN		none	The information entity has an encrypted message digest that can be used to check if the message has been tampered with. This requires a digital signature (sender's digital certificate and encrypted message digest) associated with the document entity.	
source	<pre>&lt;xsd:attributeGroup name="documentSecurity"&gt;   &lt;xsd:attribute name="isAuthenticated" default="none"&gt;     &lt;xsd:simpleType&gt;       &lt;xsd:restriction base="xsd:NMTOKEN"&gt;         &lt;xsd:enumeration value="none"/&gt;         &lt;xsd:enumeration value="transient"/&gt;         &lt;xsd:enumeration value="persistent"/&gt;         &lt;xsd:enumeration value="transient-and-persistent"/&gt;       &lt;/xsd:restriction&gt;     &lt;/xsd:simpleType&gt;   &lt;/xsd:attribute&gt;   &lt;xsd:attribute name="isConfidential" default="none"&gt;     &lt;xsd:simpleType&gt;       &lt;xsd:restriction base="xsd:NMTOKEN"&gt;         &lt;xsd:enumeration value="none"/&gt;         &lt;xsd:enumeration value="transient"/&gt;         &lt;xsd:enumeration value="persistent"/&gt;         &lt;xsd:enumeration value="transient-and-persistent"/&gt;       &lt;/xsd:restriction&gt;     &lt;/xsd:simpleType&gt;   &lt;/xsd:attribute&gt;   &lt;xsd:attribute name="isTamperProof" default="none"&gt;     &lt;xsd:simpleType&gt;</pre>					



```

<xsd:restriction base="xsd:NMTOKEN">
  <xsd:enumeration value="none"/>
  <xsd:enumeration value="transient"/>
  <xsd:enumeration value="persistent"/>
  <xsd:enumeration value="transient-and-persistent" />
</xsd:restriction>
</xsd:simpleType>
</xsd:attribute>
</xsd:attributeGroup>

```

### 8.1.32 attributeGroup name

namespace	http://www.ebxml.org/BusinessProcess				
used by	elements	<a href="#">Attachment</a> <a href="#">BinaryCollaboration</a> <a href="#">BusinessDocument</a> <a href="#">BusinessPartnerRole</a> <a href="#">BusinessTransaction</a> <a href="#">BusinessTransactionActivity</a> <a href="#">CollaborationActivity</a> <a href="#">DocumentEnvelope</a> <a href="#">Fork</a> <a href="#">Join</a> <a href="#">MultiPartyCollaboration</a> <a href="#">Package</a> <a href="#">SubstitutionSet</a> <a href="#">BusinessAction</a> <a href="#">AuthorizedRole</a>			
	complexTypees				
attributes	Name	Type	Use	Default	Annotation
	name nameID	xsd:string xsd:ID	required		
source	<xsd:attributeGroup name="name">   <xsd:attribute name="name" type="xsd:string" use="required"/>   <xsd:attribute name="nameID" type="xsd:ID"/> </xsd:attributeGroup>				

## 8.2 XML to UML cross-reference

The following is a table that references the XML element names in the XSD to their counterpart classes in the UML specification schema.

XML Element	UML Class
Attachment	Attachment
Role	AuthorizedRole
Binary Collaboration	Binary Collaboration
BusinessPartner Role	BusinessPartner Role
Business Transaction Activity	Business Transaction Activity
Business Transaction	Business Transaction
Responding BusinessActivity	Responding BusinessActivity
Requesting	Requesting

BusinessActivity	BusinessActivity
Collaboration Activity	Collaboration Activity
DocumentEnvelope	DocumentEnvelope
Documentation	None (Should be added)
ebXML Process Specification	(From Package model: ebXML Process Specification)
Failure	Failure
Include	(From Package model: Include)
MultiParty Collaboration	MultiParty Collaboration
Package	(From Package model: Package)
Performs	Performs
Schema	Schema
Fork	Fork
Start	Start
Success	Success
Join	Join
Transition	Transition
BusinessAction	BusinessAction
DocumentSecurity	DocumentSecurity

2134

2135

2136

2137

The following classes in the UML specification schema are abstract, and do not have an element equivalent in theSchema. Only their concrete subtypes are in the Schema

2138

- BusinessState

2139

- CompletionState

2140

- BusinessActivity

### 8.3 *Scoped Name Reference*

The structure of ebXML process specifications encourages re-use. An ebXMLProcessSpecification can include another ebXMLProcessSpecification by reference.

In addition the contents of a ProcessSpecification can be arranged in a recursive package structure. The ProcessSpecification is a package container, so it can contain packages within it. Package in itself is also a package container, so it can contain further packages within it.

Packages function as namespaces as per below.

Finally a Package, at any level can have PackageContent. Types of Package Content are BusinessDocument, BusinessTransaction, BinaryCollaboration, MultiPartyCollaboration.

Package Content are always uniquely named within a package. Lower level elements are uniquely named within their parent PackageContent.

Each Package Content type is a built-in context provider for the core components Logical Model for the Business Document definitions referenced by this ebXMLProcessSpecification.

Within a ebXMLProcessSpecification the following applies to naming:

Specification elements reference other specification elements by name through the use of attributes. The design pattern is that elements have a name attribute and other elements that reference the named elements do so through an attribute defined as the lowerCamelCase version of the referenced element (e.g. AuthorizedRole has attribute name while Performs, which references AuthorizedRole, has attribute authorizedRole). Two types of attributes are provided for names and references, XML ID/IDREF based and plain text. Each named element has a required name attribute and an optional nameID attribute. Referencing elements have lowerCamelCase and lowerCamelCaseIDRef attributes for the referenced element. XML ID/IDREF functionality requires all IDs to be unique within a document and that all IDREFs point to a defined ID value. Plain text attributes do not have this capability and may result in duplicate names. To unambiguously identify a referenced element using plain text attribute in the referencing attribute it is strongly recommended that XPath syntax be used. However, this is not enforced in the Schema.

The purpose of providing both solutions is to facilitate creation of Process Specification Documents directly in XML and to support future development tools that can automatically assign machine readable nameIDs and references. Both styles can be used simultaneously, in which case the ID and IDREF versions provide the unambiguous referencing and the plain text versions are used to provide meaningful names. Examples of named elements and references:

```
<Package name="ebXMLOrdering">
  <BinaryCollaboration name="OrderCollaboration" nameID="b112">
    <Role name="buyer" nameID="r224"/>
    <Role name="seller" nameID="r225"/>
  </BinaryCollaboration>
</Package>
```

```

2186
2187 <!--the XPath approach -->
2188 <Performs
2189 Role='//Package[@name="OAGOrdering"]/BinaryCollaboration[@name="OrderCollaboration"]/
2190 Role[@name="buyer"]' />
2191
2192 <!--Combination approach -->
2193 <Performs Role="buyer" RoleIDRef="r224"/>
2194
2195 It is not required to use the full path specification as shown above, other
2196 forms of XPath expressions could be used as long as they resolve to a single
2197 reference. For example if buyer was unique to the document then the XPath
2198 could have been:
2199 <Performs Role='//Role[@name="buyer"]' />
2200 Relative paths are also allowed for example:
2201 <BusinessTransactionActivity fromRole='../ Role[@name="buyer"]' ... />
2202

```

## 2203 **8.4 Sample XML document against above Schema**

2204  
2205 Provided in Appendix A  
2206

## 2207 **9 Business signal structures**

2208 The ebXML Message Service Specification signal structures provide business service  
2209 state alignment infrastructure, including unique message identifiers and digests used to  
2210 meet the basic process alignment requirements. The business signal payload structures  
2211 provided herein are optional and normative and are intended to provide business and  
2212 legal semantic to the business signals. Since signals do not differ in structure from  
2213 business transaction to business transaction, they are defined once and for all, and their  
2214 definition is implied by the conjunction of the Business Process Specification Schema  
2215 and Message Service Specification. Here are the Schema's for business signal payload  
2216 for ReceiptAcknowledgment and for AcceptanceAcknowledgement and Exception.

### 2217 **9.1.1 Signal Schema**

```

2218
2219 <?xml version="1.0" encoding="UTF-8"?>
2220 <!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
2221 -->
2222 <!-- By Himagiri Mukkamala(himagiri@sybase.com) -->
2223 <xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2224 xmlns="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS" xmlns:xlink="http://www.w3.org/1999/xlink"
2225 xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
2226 attributeFormDefault="qualified" version="2.0">
2227 <xsd:import namespace="http://www.w3.org/1999/xlink" schemaLocation="http://www.oasis-
2228 open.org/committees/ebxml-msg/schema/xlink.xsd"/>
2229 <xsd:simpleType name="non-empty-string">
2230 <xsd:restriction base="xsd:string">
2231 <xsd:minLength value="1"/>
2232 </xsd:restriction>

```

```

2233     </xsd:simpleType>
2234     <xsd:complexType name="PartyInfoType">
2235       <xsd:simpleContent>
2236         <xsd:extension base="non-empty-string">
2237           <xsd:attribute name="type" type="non-empty-string"/>
2238         </xsd:extension>
2239       </xsd:simpleContent>
2240     </xsd:complexType>
2241     <xsd:complexType name="RoleType">
2242       <xsd:attribute name="name" type="non-empty-string" use="required"/>
2243       <xsd:attributeGroup ref="xlink.grp"/>
2244     </xsd:complexType>
2245     <xsd:attributeGroup name="xlink.grp">
2246       <xsd:attribute ref="xlink:type" fixed="simple"/>
2247       <xsd:attribute ref="xlink:href" use="required"/>
2248     </xsd:attributeGroup>
2249     <xsd:complexType name="ProcessSpecificationType">
2250       <xsd:attribute name="version" type="non-empty-string"/>
2251       <xsd:attribute name="name" type="non-empty-string"/>
2252       <xsd:attributeGroup ref="xlink.grp"/>
2253       <xsd:attribute name="uuid" type="xsd:anyURI"/>
2254     </xsd:complexType>
2255   </xsd:schema>
2256

```

## 9.1.2 ReceiptAcknowledgment Signal Schema

```

2257
2258
2259
2260 <?xml version="1.0" encoding="UTF-8"?>
2261 <!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
2262 -->
2263 <!-- By Himagiri Mukkamala(himagiri@sybase.com) -->
2264 <xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2265   xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2266   xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xlink="http://www.w3.org/1999/xlink"
2267   xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
2268   attributeFormDefault="qualified" version="2.0">
2269   <xsd:include schemaLocation="Signal.xsd"/>
2270   <xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
2271     schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
2272   <xsd:element name="ReceiptAcknowledgment">
2273     <xsd:complexType>
2274       <xsd:sequence>
2275         <xsd:element name="OriginalMessageIdentifier" type="bpssignal:non-empty-string"/>
2276         <xsd:element name="OriginalDocumentIdentifier" type="bpssignal:non-empty-string"
2277           minOccurs="0"/>
2278         <xsd:element ref="bpssignal:NonRepudiationInformation" minOccurs="0"/>
2279         <xsd:element name="FromPartyInfo" type="bpssignal:PartyInfoType"/>
2280         <xsd:element name="ToPartyInfo" type="bpssignal:PartyInfoType"/>
2281         <xsd:element name="FromRole" type="bpssignal:RoleType"/>
2282         <xsd:element name="ToRole" type="bpssignal:RoleType"/>
2283         <xsd:element name="OriginalMessageDateTime" type="xsd:dateTime"/>
2284         <xsd:element name="ThisMessageDateTime" type="xsd:dateTime"/>
2285         <xsd:element name="ProcessSpecificationInfo" type="bpssignal:ProcessSpecificationType"/>
2286         <xsd:element ref="ds:Signature" minOccurs="0"/>
2287       </xsd:sequence>
2288     </xsd:complexType>
2289   </xsd:element>
2290   <xsd:element name="NonRepudiationInformation">
2291     <xsd:complexType>
2292       <xsd:sequence>
2293         <xsd:element ref="bpssignal:MessagePartNRInformation" maxOccurs="unbounded"/>
2294       </xsd:sequence>

```

```

2295         </xsd:complexType>
2296     </xsd:element>
2297     <xsd:element name="MessagePartNRInformation">
2298         <xsd:complexType>
2299             <xsd:choice>
2300                 <xsd:element name="MessagePartIdentifier" type="bpssignal:non-empty-string"/>
2301                 <xsd:element ref="ds:Reference"/>
2302             </xsd:choice>
2303         </xsd:complexType>
2304     </xsd:element>
2305 </xsd:schema>

```

### 9.1.3 AcceptanceAcknowledgement Signal Schema

```

2310 <?xml version="1.0" encoding="UTF-8"?>
2311 <!-- edited with XML Spy v4.2 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
2312 -->
2313 <!-- By Himagiri Mukkamala(himagiri@sybase.com) -->
2314 <xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2315     xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2316     xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xlink="http://www.w3.org/1999/xlink"
2317     xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
2318     attributeFormDefault="qualified" version="2.0">
2319     <xsd:include schemaLocation="Signal.xsd"/>
2320     <xsd:element name="AcceptanceAcknowledgment">
2321         <xsd:complexType>
2322             <xsd:sequence>
2323                 <xsd:element name="OriginalMessageIdentifier" type="bpssignal:non-empty-string"/>
2324                 <xsd:element name="OriginalDocumentIdentifier" type="bpssignal:non-empty-string"
2325 minOccurs="0"/>
2326                 <xsd:element name="FromPartyInfo" type="bpssignal:PartyInfoType"/>
2327                 <xsd:element name="ToPartyInfo" type="bpssignal:PartyInfoType"/>
2328                 <xsd:element name="FromRole" type="bpssignal:RoleType"/>
2329                 <xsd:element name="ToRole" type="bpssignal:RoleType"/>
2330                 <xsd:element name="OriginalMessageDateTime" type="xsd:dateTime"/>
2331                 <xsd:element name="ThisMessageDateTime" type="xsd:dateTime"/>
2332                 <xsd:element name="ProcessSpecificationInfo" type="bpssignal:ProcessSpecificationType"/>
2333             </xsd:sequence>
2334         </xsd:complexType>
2335     </xsd:element>
2336 </xsd:schema>

```

### 9.1.4 Exception Signal Schema

```

2339 <?xml version="1.0" encoding="UTF-8"?>
2340 <!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
2341 -->
2342 <!--W3C Schema generated by XML Spy v4.2 U (http://www.xmlspy.com)-->
2343 <xsd:schema targetNamespace="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2344     xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2345     xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
2346     attributeFormDefault="qualified" version="2.0">
2347     <xsd:include schemaLocation="Signal.xsd"/>
2348     <xsd:element name="Exception">
2349         <xsd:complexType>
2350             <xsd:sequence>
2351                 <xsd:element name="OriginalMessageIdentifier" type="bpssignal:non-empty-string"/>

```

```

2354      <xsd:element name="OriginalDocumentIdentifier" type="bpssignal:non-empty-string"
2355 minOccurs="0"/>
2356      <xsd:element name="Reason" type="xsd:string"/>
2357      <xsd:element name="FromPartyInfo" type="bpssignal:PartyInfoType"/>
2358      <xsd:element name="ToPartyInfo" type="bpssignal:PartyInfoType"/>
2359      <xsd:element name="FromRole" type="bpssignal:RoleType"/>
2360      <xsd:element name="ToRole" type="bpssignal:RoleType"/>
2361      <xsd:element name="OriginalMessageDateTime" type="xsd:dateTime"/>
2362      <xsd:element name="ThisMessageDateTime" type="xsd:dateTime"/>
2363      <xsd:element name="ProcessSpecificationInfo" type="bpssignal:ProcessSpecificationType"/>
2364      <xsd:element name="ExceptionType">
2365        <xsd:complexType>
2366          <xsd:choice>
2367            <xsd:element name="ReceiptException">
2368              <xsd:simpleType>
2369                <xsd:restriction base="xsd:string">
2370                  <xsd:enumeration value="Syntax"/>
2371                  <xsd:enumeration value="Authorization"/>
2372                  <xsd:enumeration value="Signature"/>
2373                  <xsd:enumeration value="Sequence"/>
2374                </xsd:restriction>
2375              </xsd:simpleType>
2376            </xsd:element>
2377            <xsd:element name="AcceptanceException">
2378              <xsd:simpleType>
2379                <xsd:restriction base="xsd:string">
2380                  <xsd:enumeration value="Business"/>
2381                  <xsd:enumeration value="Performance"/>
2382                </xsd:restriction>
2383              </xsd:simpleType>
2384            </xsd:element>
2385            <xsd:element name="GeneralException">
2386              <xsd:simpleType>
2387                <xsd:restriction base="xsd:string"/>
2388              </xsd:simpleType>
2389            </xsd:element>
2390          </xsd:choice>
2391        </xsd:complexType>
2392      </xsd:element>
2393      <xsd:element name="ExceptionMessage" type="xsd:string"/>
2394      <xsd:any namespace="##other" minOccurs="0"/>
2395    </xsd:sequence>
2396  </xsd:complexType>
2397 </xsd:element>
2398 </xsd:schema>
2399
2400
2401
2402

```

## 10 EDI support

A technical report will be made available to describe use of BPSS to describe EDI transactions.

## 11 Production Rules

This section provides a set of production rules, defining the mapping from the UML version of the *Business Process Specification Schema* to the XML version.

- 2409 The primary purpose for these production rules is to govern the one-time  
 2410 generation of the Schemaversion of the *Business Process Specification Schema*  
 2411 from the UML Class Diagram version of *Business Process Specification Schema*.
- 2412 The Class Diagram version of *Business Process Specification Schema* is not  
 2413 intended for the direct creation of ebXML Business Process Specifications.  
 2414 However, if a *Business Process Specification* was in fact (programmatically)  
 2415 created as an instance of this class diagram, the production rules would also  
 2416 provide the prescriptive definition necessary to translate a such an instance into  
 2417 a XML Specification Document conformant with theSchema. The production  
 2418 rules are defined for concrete classes, abstract classes, aggregate associations,  
 2419 specialization associations and unidirectional associations.
- 2420 1. Classes are rendered as XML elements.
  - 2421 2. Class attributes are rendered as XML attributes. NOTE: occurrence  
 2422 requirements (required vs optional) and default values for attributes are not  
 2423 modeled.
  - 2424 3. Specialization classes (classes that inherit from another class) are rendered  
 2425 as XML elements including all attributes and aggregate associations from the  
 2426 base class. Repeated attributes are normalized to a single occurrence.
  - 2427 4. Abstract classes are not rendered in the XML Schema. Abstract classes are  
 2428 inherited from and represent a form of collection. A class that aggregates an  
 2429 abstract class, essentially aggregates “any of each” of the specialization  
 2430 classes.
  - 2431 5. An aggregate association renders the aggregated class as an XML child  
 2432 element with appropriate cardinality.
  - 2433 6. A unidirectional association defines an attribute in the originating class of the  
 2434 same name as the class the association points to. This type of attribute is  
 2435 called a “reference attribute” and contains the name of the class it points to.  
 2436 The referenced class must have a “name” attribute.
  - 2437 7. A class attribute data type, that has a class of the same name with stereotype  
 2438 <<Enumeration>> is rendered as an XML attribute enumeration. The  
 2439 Enumeration class does not have an explicit association.
  - 2440 8. A class attribute data type (e.g. Time, URI, Boolean) that has no  
 2441 corresponding class definition is rendered as a string in theSchema. In the  
 2442 XML Schema version these data types are mapped as:  
 2443 Time - xsd:duration  
 2444 URI - xsd:anyURI  
 2445 Boolean - xsd:boolean
  - 2446 9. Each class is given an optional “Documentation\*” element which is intended  
 2447 for annotation of the specification instances. This is not modeled.
- 2448



## Appendix A: Sample XML Business Process Specification

```

2451 <ProcessSpecification name="Simple" version="1.1" uuid="Simple-2434134"
2452 xmlns="http://www.ebxml.org/BusinessProcess" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2453 xsi:schemaLocation="http://www.ebxml.org/BusinessProcess
2454 ebBPSS1.04.xsd.xml">
2455   <!-- Business Documents -->
2456   <BusinessDocument name="Catalog Request" specificationLocation="http://www.yyx.com/CatalogReq.xsd"/>
2457   <BusinessDocument name="Catalog" specificationLocation="http://www.yyx.com/Catalog.xsd"/>
2458   <BusinessDocument name="Purchase Order" specificationLocation="http://www.yyx.com/PO.xsd"/>
2459   <BusinessDocument name="PO Acknowledgement" specificationLocation="http://www.yyx.com/POAck.xsd"/>
2460   <BusinessDocument name="Credit Request" specificationLocation="http://www.yyx.com/CreditReq.xsd"/>
2461   <BusinessDocument name="Credit Confirm" specificationLocation="http://www.yyx.com/CreditCon.xsd"/>
2462   <BusinessDocument name="ASN" specificationLocation="http://www.yyx.com/CatalogASN.xsd"/>
2463   <BusinessDocument name="CreditAdvice" specificationLocation="http://www.yyx.com/CreditAdvice.xsd"/>
2464   <BusinessDocument name="DebitAdvice" specificationLocation="http://www.yyx.com/DebitAdvice.xsd"/>
2465   <BusinessDocument name="Invoice" specificationLocation="http://www.yyx.com/Invoice.xsd"/>
2466   <BusinessDocument name="Payment" specificationLocation="http://www.yyx.com/Payment.xsd"/>
2467   <BusinessDocument name="Inventory Report Request"
specificationLocation="http://www.yyx.com/InvReq.xsd"/>
2469   <BusinessDocument name="Inventory Report" specificationLocation="http://www.yyx.com/InvRep.xsd"/>
2470   <Package name="Ordering">
2471
2472     <!-- Here are all the Business Transactions needed -->
2473     <BusinessTransaction name="Catalog Request">
2474       <RequestingBusinessActivity name="RequestCatalog">
2475         <DocumentEnvelope businessDocument="Catalog Request"/>
2476       </RequestingBusinessActivity>
2477       <RespondingBusinessActivity name="SendCatalog">
2478         <DocumentEnvelope isPositiveResponse="true" businessDocument="Catalog"/>
2479       </RespondingBusinessActivity>
2480     </BusinessTransaction>
2481
2482     <BusinessTransaction name="Create Order">
2483       <RequestingBusinessActivity name="SendOrder" isNonRepudiationRequired="true"
timeToAcknowledgeReceipt="P2D" timeToAcknowledgeAcceptance="P3D">
2484         <DocumentEnvelope businessDocument="Purchase Order"/>
2485       </RequestingBusinessActivity>
2486       <RespondingBusinessActivity name="SendPOAcknowledgement" isNonRepudiationRequired="true"
timeToAcknowledgeReceipt="P5D">
2487         <DocumentEnvelope isPositiveResponse="true" businessDocument="PO Acknowledgement"/>
2488       </RespondingBusinessActivity>
2489     </BusinessTransaction>
2490
2491     <BusinessTransaction name="Check Credit ">
2492       <RequestingBusinessActivity name="CreditCheck">
2493         <DocumentEnvelope businessDocument="Credit Request"/>
2494       </RequestingBusinessActivity>
2495       <RespondingBusinessActivity name="ConfirmCredit">
2496         <DocumentEnvelope isPositiveResponse="true" businessDocument="Credit Confirm"/>
2497       </RespondingBusinessActivity>
2498     </BusinessTransaction>
2499
2500     <BusinessTransaction name="Notify of advance shipment">
2501       <RequestingBusinessActivity name="AdvanceShipmentNotification">
2502         <DocumentEnvelope businessDocument="ASN"/>
2503       </RequestingBusinessActivity>
2504       <RespondingBusinessActivity name="ASNResponse" />
2505     </BusinessTransaction>
2506
2507     <BusinessTransaction name="Process Credit Payment">

```

```

2510     <RequestingBusinessActivity name="CreditPaymentProcess">
2511       <DocumentEnvelope businessDocument="CreditAdvice"/>
2512     </RequestingBusinessActivity>
2513     <RespondingBusinessActivity name="CreditPaymentProcessResponse">
2514       <DocumentEnvelope isPositiveResponse="true" businessDocument="DebitAdvice"/>
2515     </RespondingBusinessActivity>
2516   </BusinessTransaction>
2517
2518   <BusinessTransaction name="Process Payment">
2519     <RequestingBusinessActivity name="PaymentProcess">
2520       <DocumentEnvelope businessDocument="Invoice"/>
2521     </RequestingBusinessActivity>
2522     <RespondingBusinessActivity name="SendPayment">
2523       <DocumentEnvelope isPositiveResponse="true" businessDocument="Payment"/>
2524     </RespondingBusinessActivity>
2525   </BusinessTransaction>
2526
2527   <BusinessTransaction name="Request Inventory Report">
2528     <RequestingBusinessActivity name="">
2529       <DocumentEnvelope businessDocument="Inventory Report Request"/>
2530     </RequestingBusinessActivity>
2531     <RespondingBusinessActivity name="Inventory Report">
2532       <DocumentEnvelope businessDocument="Inventory Report"/>
2533     </RespondingBusinessActivity>
2534   </BusinessTransaction>
2535
2536   <!-- Now the Binary Collaborations -->
2537   <BinaryCollaboration name="Request Catalog" initiatingRoleID="1122B1">
2538     <Role name="requestor" nameID="1122B1"/>
2539     <Role name="provider" nameID="2211A1"/>
2540     <Start toBusinessState="Catalog Request"/>
2541     <BusinessTransactionActivity name="Catalog Request" businessTransaction="Catalog Request"
2542 fromRole="requestor" toRole="provider"/>
2543     <Success fromBusinessState="Catalog Request" conditionGuard="Success"/>
2544     <Failure fromBusinessState="Catalog Request" conditionGuard="Failure"/>
2545   </BinaryCollaboration>
2546
2547   <BinaryCollaboration name="Firm Order" timeToPerform="P2D" initiatingRoleID="1122B2">
2548     <Documentation>timeToPerform = Period: 2 days from start of transaction</Documentation>
2549     <Role name="buyer" nameID="1122B2"/>
2550     <Role name="seller" nameID="1122B3"/>
2551     <Start toBusinessState="Create Order"/>
2552     <BusinessTransactionActivity name="Create Order" businessTransaction="Create Order"
2553 fromRole="buyer" toRole="seller"/>
2554     <Success fromBusinessState="Create Order" conditionGuard="Success"/>
2555     <Failure fromBusinessState="Create Order" conditionGuard="Failure"/>
2556   </BinaryCollaboration>
2557
2558   <BinaryCollaboration name="Product Fulfillment" timeToPerform="P5D" initiatingRoleID="1122B2">
2559     <Documentation>timeToPerform = Period: 5 days from start of transaction</Documentation>
2560     <Role name="buyer" nameID="1122B2"/>
2561     <Role name="seller" nameID="1122B3"/>
2562     <Start toBusinessState="Create Order"/>
2563     <BusinessTransactionActivity name="Create Order" businessTransaction="Create Order"
2564 fromRole="buyer" toRole="seller"/>
2565     <BusinessTransactionActivity name="Notify shipment" businessTransaction="Notify of advance
2566 shipment" fromRole="seller" toRole="buyer"/>
2567     <Transition fromBusinessState="Create Order" toBusinessState="Notify shipment"/>
2568     <Success fromBusinessState="Notify shipment" conditionGuard="Success"/>
2569     <Failure fromBusinessState="Notify shipment" conditionGuard="Failure"/>
2570   </BinaryCollaboration>
2571
2572   <BinaryCollaboration name="Inventory Status" initiatingRoleID="1122B1">
2573     <Role name="requestor" nameID="1122B1"/>

```

```

2574         <Role name="provider" nameID="2211A1"/>
2575         <Start toBusinessState="Inventory Report Request"/>
2576         <BusinessTransactionActivity name="Inventory Report Request" businessTransaction="Inventory
Report Request" fromRole="requestor" toRole="provider"/>
2577         <Success fromBusinessState="Inventory Report Request" conditionGuard="Success"/>
2578         <Failure fromBusinessState="Inventory Report Request" conditionGuard="Failure"/>
2579     </BinaryCollaboration>
2580
2581     <BinaryCollaboration name="Credit Inquiry" initiatingRoleID="9122B1">
2582         <Role name="creditor" nameID="9122B1"/>
2583         <Role name="credit service" nameID="8122B1"/>
2584         <Start toBusinessState="Check Credit"/>
2585         <BusinessTransactionActivity name="Check Credit" businessTransaction="Check Credit"
fromRole="creditor" toRole="credit service"/>
2586         <Success fromBusinessState="Check Credit" conditionGuard="Success"/>
2587         <Failure fromBusinessState="Check Credit" conditionGuard="Failure"/>
2588     </BinaryCollaboration>
2589
2590     <BinaryCollaboration name="Credit Payment" initiatingRoleID="6122B1">
2591         <Role name="payee" nameID="6122B1"/>
2592         <Role name="payor" nameID="7122B1"/>
2593         <Start toBusinessState="Process Credit Payment"/>
2594         <BusinessTransactionActivity name="Process Credit Payment" businessTransaction="Process Credit
Payment" fromRole="payee" toRole="payor"/>
2595         <Success fromBusinessState="Process Credit Payment" conditionGuard="Success"/>
2596         <Failure fromBusinessState="Process Credit Payment" conditionGuard="Failure"/>
2597     </BinaryCollaboration>
2598
2599     <!-- A compound BinaryCollaboration for illustration purposes-->
2600     <BinaryCollaboration name="Credit Charge" initiatingRoleID="8132B1">
2601         <Role name="charger" nameID="8132B1"/>
2602         <Role name="credit service" nameID="8122B1"/>
2603         <Start toBusinessState="Credit Inquiry"/>
2604         <CollaborationActivity name="Credit Inquiry" binaryCollaboration="Credit Inquiry" fromRole="charger"
toRole="credit service"/>
2605         <CollaborationActivity name="Credit Payment" binaryCollaboration="Credit Payment"
fromRole="charger" toRole="payor"/>
2606         <Transition fromBusinessState="Credit Inquiry" toBusinessState="Credit Payment"/>
2607         <Success fromBusinessState="Credit Payment" conditionGuard="Success"/>
2608         <Failure fromBusinessState="Credit Payment" conditionGuard="Failure"/>
2609     </BinaryCollaboration>
2610
2611     <BinaryCollaboration name="Fulfillment Payment" initiatingRoleID="6122B1">
2612         <Role name="payee" nameID="6122B1"/>
2613         <Role name="payor" nameID="7122B1"/>
2614         <Start toBusinessState="Process Payment"/>
2615         <BusinessTransactionActivity name="Process Payment" businessTransaction="Process Payment"
fromRole="payee" toRole="payor"/>
2616         <Success fromBusinessState="Process Payment" conditionGuard="Success"/>
2617         <Failure fromBusinessState="Process Payment" conditionGuard="Failure"/>
2618     </BinaryCollaboration>
2619
2620     <!-- First the overall MultiParty Collaboration -->
2621     <MultiPartyCollaboration name="DropShip">
2622         <BusinessPartnerRole name="Customer">
2623             <Performs role="requestor" roleIDRef="1122B1"/>
2624             <Performs role="buyer" roleIDRef="1122B2"/>
2625             <Transition fromBusinessState="Catalog Request" toBusinessState="Create Order"/>
2626         </BusinessPartnerRole>
2627         <BusinessPartnerRole name="Retailer">
2628             <Performs role="provider" roleIDRef="2211A1"/>
2629             <Performs role="seller" roleIDRef="1122B3"/>
2630             <Performs role="creditor" roleIDRef="9122B1"/>
2631             <Performs role="buyer" roleIDRef="1122B2"/>

```

```

2638         <Performs role="payee" roleIDRef="6122B1"/>
2639         <Performs role="payor" roleIDRef="7122B1"/>
2640         <Performs role="requestor" roleIDRef="1122B1"/>
2641         <Transition fromBusinessState="Create Order" toBusinessState="Check Credit"/>
2642         <Transition fromBusinessState="Check Credit" toBusinessState="Credit Payment"/>
2643     </BusinessPartnerRole>
2644     <BusinessPartnerRole name="DropShip Vendor">
2645         <Performs role="seller" roleIDRef="1122B3"/>
2646         <Performs role="payee" roleIDRef="6122B1"/>
2647         <Performs role="provider" roleIDRef="2211A1"/>
2648     </BusinessPartnerRole>
2649     <BusinessPartnerRole name="Credit Authority">
2650         <Performs role="credit service" roleIDRef="8122B1"/>
2651         <Performs role="payor" roleIDRef="7122B1"/>
2652     </BusinessPartnerRole>
2653 </MultiPartyCollaboration>
2654 </Package>
2655 </ProcessSpecification>
2656
2657

```

## Appendix B: Sample XML Signal

```

2659 <?xml version="1.0" encoding="UTF-8"?>
2660 <!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by Himagiri Mukkamala (Web Services Architecture WG)
2661 -->
2662 <!--Sample XML file generated by XML Spy v4.2 U (http://www.xmlspy.com)-->
2663 <bpssignal:Exception xmlns:bpssignal="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS"
2664 xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
2665 xsd:schemaLocation="http://www.ebxml.org/BusinessProcess/BPSS_SIGNALS
2666 ExceptionSignal.xsd">
2667     <bpssignal:OriginalMessageIdentifier>MessageIdentifier-1</bpssignal:OriginalMessageIdentifier>
2668     <bpssignal:Reason>State transition failure</bpssignal:Reason>
2669     <bpssignal:FromPartyInfo bpssignal:type="DUNS.com">PartyA</bpssignal:FromPartyInfo>
2670     <bpssignal:ToPartyInfo bpssignal:type="DUNS.com">PartyB</bpssignal:ToPartyInfo>
2671     <bpssignal:FromRole bpssignal:name="Buyer" xlink:type="simple"
2672     xlink:href="http://www.rosettanet.org/processes/3A4.xml#Buyer"/>
2673     <bpssignal:ToRole bpssignal:name="Seller" xlink:type="simple"
2674     xlink:href="http://www.rosettanet.org/processes/3A4.xml#Seller"/>
2675     <bpssignal:OriginalMessageDateTime>2002-03-05T19:00:00</bpssignal:OriginalMessageDateTime>
2676     <bpssignal:ThisMessageDateTime>2002-03-05T20:00:00</bpssignal:ThisMessageDateTime>
2677     <bpssignal:ProcessSpecificationInfo bpssignal:version="2.0"
2678     bpssignal:name="PIP3A4RequestPurchaseOrder" xlink:type="simple"
2679     xlink:href="http://www.rosettanet.org/processes/3A4.xml" bpssignal:uuid="urn:icann:rosettanet.org:bpid:3A4$2.0"/>
2680     <bpssignal:ExceptionType>
2681         <bpssignal:ReceiptException>Signature</bpssignal:ReceiptException>
2682     </bpssignal:ExceptionType>
2683     <bpssignal:ExceptionMessage>Signature Validation Failed for request
2684     message</bpssignal:ExceptionMessage>
2685 </bpssignal:Exception>
2686
2687
2688
2689
2690
2691
2692
2693
2694

```

## 12 References

1. UN/CEFACT Modeling Methodology (UMM) as defined in the N090R10 specification, <http://www.gefeg.com/tmwg/n090r10.htm>
2. ebXML Technical Architecture Specification, version 1.04, <http://www.ebxml.org/specs/ebTA.pdf>
3. ebXML Core Components Dictionary, version 1.04, <http://www.ebxml.org/specs/ccDICT.pdf>
4. ebXML Naming Convention for Core Components, version 1.04, <http://www.ebxml.org/specs/ebCCNAM.pdf>
5. ebXML Business Process and Business Information Analysis Overview, version 1.0, <http://www.ebxml.org/specs/bpOVER.pdf>
6. ebXML Business Process Analysis Worksheets & Guidelines, version 1.0, <http://www.ebxml.org/specs/bpWS.pdf>
7. ebXML E-Commerce Patterns, version 1.0, <http://www.ebxml.org/specs/bpPATT.pdf>
8. ebXML Catalog of Common Business Processes, version 1.0, <http://www.ebxml.org/specs/bpPROC.pdf>
9. RosettaNet Implementation Framework: Core Specification, Version: Release 2.00.00, 13 July 2001
10. Key Words for use in RFCs to Indicate Requirement Levels, Internet Engineering Task Force RFC 2119, <http://www.ietf.org/rfc/rfc2119.txt>.
11. Extensible Markup Language (XML), World Wide Web Consortium, <http://www.w3.org/XML>.
12. XML Schema Part 1: Structures, Worldwide Web Consortium, <http://www.w3.org/TR/xmlschema-1/>.
13. XML Schema Part 2: Datatypes, Worldwide Web Consortium, <http://www.w3.org/TR/xmlschema-2/>.
14. ebXML Message Service Specification, [http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS\\_v2\\_0.pdf](http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf).
15. ebXML Registry Services Specification, <http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf>.
16. ebXML Collaboration-Protocol Profile and Agreement Specification V1.9, [http://www.oasis-open.org/committees/ebxml-cppa/documents/working\\_drafts/ebCPP-1\\_9.pdf](http://www.oasis-open.org/committees/ebxml-cppa/documents/working_drafts/ebCPP-1_9.pdf)

## 13 Disclaimer

The views and specification expressed in this document are those of the authors and are not necessarily those of their employers. The authors and their employers specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this design.



## 14 Contact Information

**Team Leader (Of the BP team):**

Brian Hayes

Collaborative Domain

Tel: (925) 788-6303

<mailto:brian.hayes@UCLAlumni.net>**Editor (of this document):**

Pallavi Malu (Project Editor)

Intel Corporation

5000 W.Chandler Blvd,

Chandler, AZ 85226

Tel: 480-552-0463

<mailto:pallavi.g.malu@intel.com>

Jean-Jacques\_Dubray, (Editor – Sections 1-7)

**Eigner** *Precision Lifecycle Management*

200 Fifth Avenue

Waltham, MA 02451

Tel: 781-472-6317

[jjd@eigner.com](mailto:jjd@eigner.com)

2763

2764 Copyright Statement

2765 Copyright © UN/CEFACT and OASIS, 2001. All Rights Reserved.

2766

2767 This document and translations of it may be copied and furnished to others, and  
2768 derivative works that comment on or otherwise explain it or assist in its implementation  
2769 may be prepared, copied, published and distributed, in whole or in part, without  
2770 restriction of any kind, provided that the above copyright notice and this paragraph are  
2771 included on all such copies and derivative works. However, this document itself may not  
2772 be modified in any way, such as by removing the copyright notice or references to  
2773 ebXML, UN/CEFACT, or OASIS, except as required to translate it into languages other  
2774 than English.

2775

2776 The limited permissions granted above are perpetual and will not be revoked by ebXML  
2777 or its successors or assigns. This document and the information contained herein is  
2778 provided on an "AS IS" basis and ebXML DISCLAIMS ALL WARRANTIES,  
2779 EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY  
2780 THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY  
2781 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR  
2782 FITNESS FOR A PARTICULAR PURPOSE

2783